For Windows Systems

Nonstop Database

HiRDB Version 9

System Definition

3020-6-453-40(E)

■ Relevant program products

List of program products:

For the Windows Server 2003 x64 Editions, Windows Server 2008 R2, Windows Server 2008 (x64), Windows Server 2012, Windows XP x64 Edition, Windows Vista Ultimate (x64), Windows Vista Business (x64), Windows Vista Enterprise (x64), Windows 7 Professional (x64), Windows 7 Enterprise (x64), Windows 7 Ultimate (x64), Windows 8 Pro (x64), and Windows 8 Enterprise (x64) operating systems:

P-2962-9197 HiRDB Server Version 9 09-04

For the Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2012, Windows 7, and Windows 8 operating systems:

P-2662-1197 HiRDB/Run Time Version 9 09-04

For the Windows XP x64 Edition, Windows Server 2003 x64 Editions, Windows Vista (x64), Windows Server 2008 (x64), Windows Server 2012, Windows 7 (x64), Windows 8 Pro (x64), and Windows 8 Enterprise (x64) operating systems:

P-2962-1197 HiRDB/Run Time Version 9(64) 09-04

This edition of the manual is released for the preceding program products, which have been developed under a quality management system that has been certified to comply with ISO9001 and TickIT. This manual can be used for products other than the products shown above. For details, see the *Release Notes*.

■ Trademarks

■ Restrictions

■ Issued

Mar. 2014: 3020-6-453-40(E)

■ Copyright

Preface

This manual describes the system definitions for HiRDB Version 9, a nonstop database server program product.

■ Intended readers

This manual is intended for users who will be constructing or operating *HiRDB Version 9* ("HiRDB") relational database systems.

Readers of this manual must have the following:

- A basic knowledge of managing Windows
- A basic knowledge of SQL

This manual is based on the following manuals, which we recommend that you read before reading this one:

- *HiRDB Version 9 Installation and Design Guide*
- *HiRDB Version 9 System Operation Guide*

■ Organization of this manual

This manual is organized into the following chapters:

1. Overview

Chapter 1 explains the organization and syntax rules of the HiRDB system definitions. This chapter also provides a list of the operands.

2. System Common Definition

Chapter 2 explains the system common definition.

3. Unit Control Information Definition

Chapter 3 explains the unit control information definition.

4. Server Common Definition

Chapter 4 explains the server common definition.

5. Single Server Definition

Chapter 5 explains the single server definition.

6. Front-End Server Definition

Chapter 6 explains the front-end server definition.

7. Dictionary Server Definition

Chapter 7 explains the dictionary server definition.

8. Back-End Server Definition

Chapter 8 explains the back-end server definition.

9. UAP Environment Definition

Chapter 9 explains the UAP environment definition.

A. Operand Specification Values

Appendix A explains the information to be specified in operands (arranged by types of operands).

B. Examples of Definitions

Appendix B provides examples of HiRDB system definitions.

C. Formulas for Determining Operand Specification Values

Appendix C provides and explains formulas that can be used to determine operand specification values.

D. Determining the Number of Locked Resources

Appendix D provides and explains formulas that can be used to determine the numbers of locked resources.

E. Operands Checked by the pdconfchk Command

Appendix E provides a list of operands that can be checked by executing the `pdconfchk` command.

F. List of Operands That Can Be Specified When Using the Standby-less System Switchover (Effects Distributed) Facility (Unit Control Information Definition)

Appendix F lists the operands that can be specified in the unit control information definition for the standby-less system switchover (effects distributed) facility.

■ Related publications

This manual is part of a related set of manuals. The manuals in the set are listed below (with the manual numbers):

HiRDB (for Windows)

- *HiRDB Version 9 System Description* (3020-6-451(E)), for Windows systems
- *HiRDB Version 9 Installation and Design Guide* (3020-6-452(E)), for Windows systems
- *HiRDB Version 9 System Operation Guide* (3020-6-454(E)), for Windows systems
- *HiRDB Version 9 Command Reference* (3020-6-455(E)), for Windows systems
- *HiRDB Version 8 Batch Job Accelerator* (3020-6-368)#

HiRDB (for UNIX)

- *HiRDB Version 9 Description* (3000-6-451(E)), for UNIX systems
- *HiRDB Version 9 Installation and Design Guide* (3000-6-452(E)), for UNIX systems
- *HiRDB Version 9 System Definition* (3000-6-453(E)), for UNIX systems
- *HiRDB Version 9 System Operation Guide* (3000-6-454(E)), for UNIX systems
- *HiRDB Version 9 Command Reference* (3000-6-455(E)), for UNIX systems
- *HiRDB Version 9 Staticizer Option Description and User's Guide* (3000-6-463), for UNIX systems#
- *HiRDB Version 9 Disaster Recovery System Configuration and Operation Guide* (3000-6-464(E)), for UNIX systems
- *HiRDB Version 9 Batch Job Accelerator* (3020-6-468)#
- *HiRDB Version 9 Memory Database Installation and Operation Guide* (3020-6-469), for UNIX systems#1

HiRDB (for both Windows and UNIX)

- *HiRDB Version 9 UAP Development Guide* (3020-6-456(E))
- *HiRDB Version 9 SQL Reference* (3020-6-457(E))
- *HiRDB Version 9 Messages* (3020-6-458(E))
- *HiRDB Version 9 XDM/RD E2 Connection Facility* (3020-6-465)#
- *HiRDB Version 9 XML Extension* (3020-6-480)#
- *HiRDB Version 9 Text Search Plug-in* (3020-6-481)#
- *HiRDB Version 8 Security Guide* (3020-6-359)#
- *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide* (3020-6-360(E))
- *HiRDB Datareplicator Extension Version 8* (3020-6-361)#
- *HiRDB Dataextractor Version 8 Description, User's Guide and Operator's Guide* (3020-6-362(E))

In references to HiRDB Version 9 manuals, this manual omits the phrases *for UNIX systems* and *for Windows systems*. Refer to either the UNIX or Windows HiRDB manual, whichever is appropriate for your platform.

Others

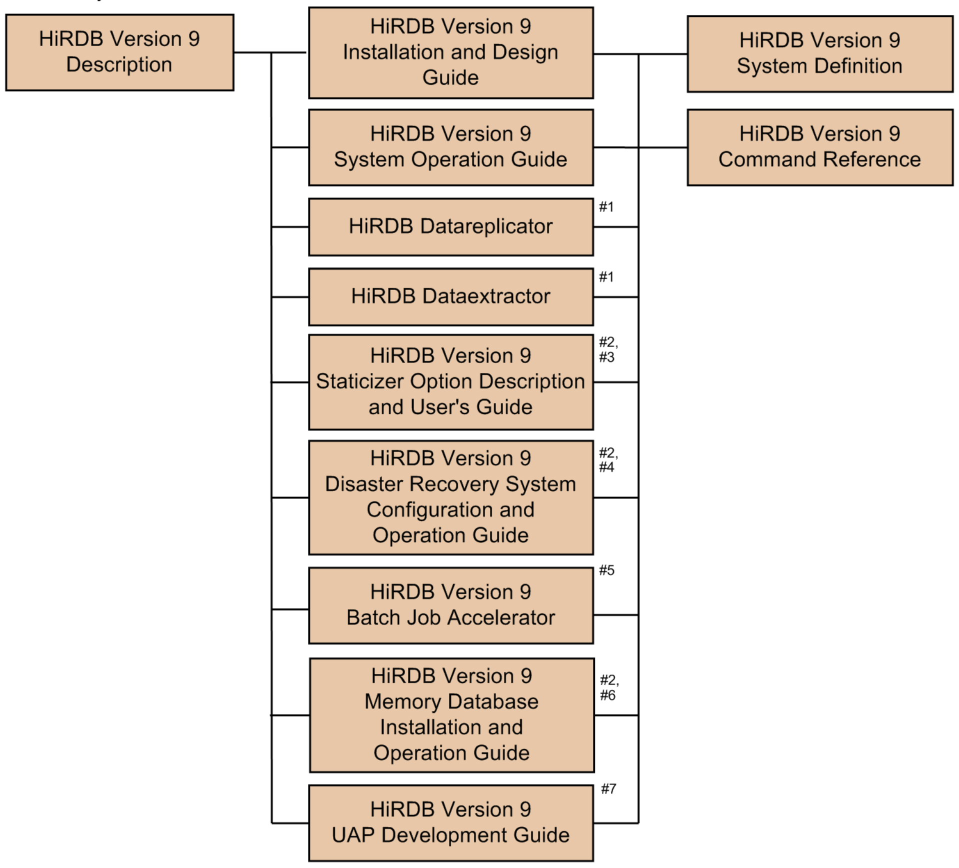
- *Hitachi HA Toolkit* (3000-9-115)#

#: This manual has been published in Japanese only; it is not available in English.

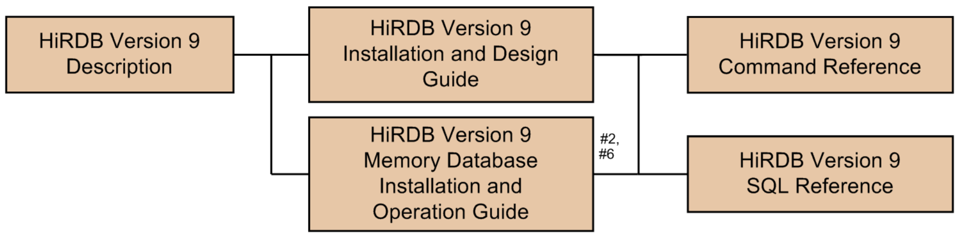
■ Organization of HiRDB manuals

The HiRDB manuals are organized as shown below. For the most efficient use of these manuals, it is suggested that they be read in the order they are shown, going from left to right.

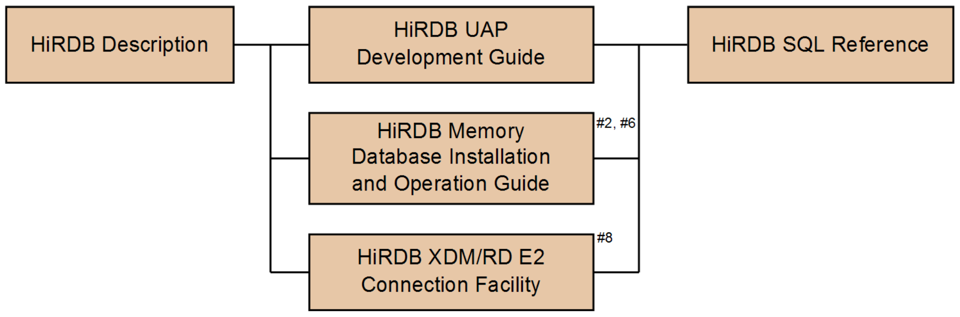
Manuals for system administrators:


HiRDB Version 9 Description
HiRDB Version 9 Installation and Design Guide
HiRDB Version 9 System Operation Guide
HiRDB Datareplicator #1
HiRDB Dataextractor #1
HiRDB Version 9 Staticizer Option Description and User's Guide #2, #3
HiRDB Version 9 Disaster Recovery System Configuration and Operation Guide #2, #4
HiRDB Version 9 Batch Job Accelerator #5
HiRDB Version 9 Memory Database Installation and Operation Guide #2, #6
HiRDB Version 9 UAP Development Guide #7
HiRDB Version 9 System Definition
HiRDB Version 9 Command Reference

Manuals for users who create tables:


HiRDB Version 9 Description
HiRDB Version 9 Installation and Design Guide
HiRDB Version 9 Memory Database Installation and Operation Guide #2, #6
HiRDB Version 9 Command Reference
HiRDB Version 9 SQL Reference

For users who create or execute UAPs:


HiRDB Description
HiRDB UAP Development Guide
HiRDB SQL Reference
HiRDB Memory Database Installation and Operation Guide
#2, #6
HiRDB XDM/RD E2 Connection Facility
#8

#1: Read if you intend to use the replication facility to link data.
#2: Published for UNIX only. There is no corresponding Windows manual.
#3: Read if you intend to use the inner replica facility.
#4: Read if you intend to configure a disaster recovery system.
#5: Read if you intend to use in-memory data processing to accelerate batch operations.
#6: Read if you intend to use the memory database facility.
#7: Read if you intend to link HiRDB to an OLTP system.
#8: Read if you intend to use the XDM/RD E2 connection facility to perform operations on XDM/RD E2 databases.

■ Conventions: Abbreviations

Unless otherwise required, this manual uses the following abbreviations for product and other names:

Full name or meaning	Abbreviation	
HiRDB Server Version 9	HiRDB/Single Server	HiRDB or HiRDB Server
	HiRDB/Parallel Server	
HiRDB/Developer's Kit Version 9	HiRDB/Developer's Kit	HiRDB Client
HiRDB/Developer's Kit Version 9(64)		
HiRDB/Run Time Version 9	HiRDB/Run Time	
HiRDB/Run Time Version 9(64)		
HiRDB Accelerator Version 8	HiRDB Accelerator	
HiRDB Accelerator Version 9		
HiRDB Adapter for XML - Standard Edition	HiRDB Adapter for XML	
HiRDB Adapter for XML - Enterprise Edition		
HiRDB Advanced High Availability Version 9	HiRDB Advanced High Availability	
HiRDB Control Manager	HiRDB CM	
HiRDB Control Manager Agent	HiRDB CM Agent	
HiRDB Dataextractor Version 8	HiRDB Dataextractor	
HiRDB Datareplicator Version 8	HiRDB Datareplicator	
HiRDB Disaster Recovery Light Edition Version 9	HiRDB Disaster Recovery Light Edition	
HiRDB Non Recover Front End Server Version 9	HiRDB Non Recover FES	
HiRDB Staticizer Option Version 9	HiRDB Staticizer Option	
HiRDB Text Search Plug-in Version 9	HiRDB Text Search Plug-in	
HiRDB XML Extension Version 9	HiRDB XML Extension	

Full name or meaning	Abbreviation
Single server	SDS
System manager	MGR
Front-end server	FES
Dictionary server	DS
Back-end server	BES
Microsoft(R) ActiveX(R)	ActiveX
DNCWARE ClusterPerfect (Linux Edition)	ClusterPerfect
DataStage(R)	DataStage
DB2 Universal Database for OS/390 Version 6	DB2
JP1/Magnetic Tape Access	EasyMT
EasyMT	
JP1/Automatic Job Management System 2 - Scenario Operation	JP1/AJS2-SO
JP1/Automatic Job Management System 3	JP1/AJS3
JP1/Automatic Job Management System 2	
JP1/Cm2/Extensible SNMP Agent	JP1/ESA
JP1/Cm2/Extensible SNMP Agent for Mib Runtime	
JP1/Integrated Management - Manager	JP1/Integrated Management or JP1/IM
JP1/Integrated Management - View	
JP1/NETM/Audit - Manager	JP1/NETM/Audit
JP1/NETM/DM	JP1/NETM/DM
JP1/NETM/DM Manager	
JP1/Cm2/Network Node Manager	JP1/NNM
JP1/Performance Management	JP1/PFM
JP1/Performance Management - Agent Option for HiRDB	JP1/PFM-Agent for HiRDB
JP1/Performance Management - Agent Option for Platform	JP1/PFM-Agent for Platform
JP1/Performance Management/SNMP System Observer	JP1/SSO
JP1/VERITAS NetBackup BS V4.5 Agent for HiRDB License	JP1/VERITAS NetBackup Agent for HiRDB License
JP1/VERITAS NetBackup V4.5 Agent for HiRDB License	
JP1/VERITAS NetBackup 5 Agent for HiRDB License	
LifeKeeper for Linux V7 Update1	LifeKeeper
MetaBroker(R)	MetaBroker
MetaStage(R)	MetaStage
Microsoft(R) Office Excel	Microsoft Excel or Excel
JP1/Magnetic Tape Library	MTguide
JP1/VERITAS NetBackup BS v4.5	NetBackup

Full name or meaning	Abbreviation	
JP1/VERITAS NetBackup v4.5		
PowerHA for AIX, V5.5	PowerHA	
PowerHA SystemMirror V6.1		
QualityStage(TM)	QualityStage	
OpenTP1/Server Base Enterprise Option	TP1/EE	
Hitachi TrueCopy	TrueCopy	
Hitachi TrueCopy Asynchronous		
Hitachi TrueCopy basic		
Hitachi TrueCopy Software		
TrueCopy		
TrueCopy Asynchronous		
TrueCopy remote replicator		
Hitachi Universal Replicator Software	Universal Replicator	
Universal Replicator		
Microsoft$^{(R)}$ Visual C++$^{(R)}$	Visual C++ or C++ language	
Oracle WebLogic Server	WebLogic Server	
Virtual-storage Operating System 3/Forefront System Product	VOS3/FS	VOS3
Virtual-storage Operating System 3/Leading System Product	VOS3/LS	
Virtual-storage Operating System 3/Unific System Product	VOS3/US	
VOS3 Database Connection Server	DB Connection Server	
Extensible Data Manager/Base Extended Version 2 XDM Basic Program XDM/BASE E2	XDM/BASE E2	
XDM/Data Communication and Control Manager 3 XDM Data Communication Management System XDM/DCCM3	XDM/DCCM3	
XDM/Relational Database Relational Database System XDM/RD	XDM/RD	XDM/RD
XDM/Relational Database Extended Version 2 Relational Database System XDM/RD E2	XDM/RD E2	
HP-UX 11i V2 (IPF)	HP-UX or HP-UX (IPF)	
HP-UX 11i V3 (IPF)		
AIX 5L V5.2	AIX 5L	AIX
AIX 5L V5.3		
AIX V6.1	AIX V6.1	
AIX V7.1	AIX V7.1	
Linux$^{(R)}$	Linux	
Red Hat Enterprise Linux$^{(R)}$ AS 4(AMD64 & Intel EM64T)	Linux AS 4	Linux

Full name or meaning	Abbreviation	
Red Hat Enterprise Linux(R) AS 4(x86)		
Red Hat Enterprise Linux(R) ES 4(AMD64 & Intel EM64T)	Linux ES 4	
Red Hat Enterprise Linux(R) ES 4(x86)		
Red Hat Enterprise Linux(R) 5 Advanced Platform (x86)	Linux 5	
Red Hat Enterprise Linux(R) 5 (x86)		
Red Hat Enterprise Linux(R) 5 Advanced Platform (AMD/Intel 64)		
Red Hat Enterprise Linux(R) 5 (AMD/Intel 64)		
Red Hat Enterprise Linux(R) 6 (32-bit x86)	Linux 6	
Red Hat Enterprise Linux(R) 6 (64-bit x86_64)		
Red Hat Enterprise Linux(R) AS 4(AMD64 & Intel EM64T)	Linux (EM64T)	
Red Hat Enterprise Linux(R) ES 4(AMD64 & Intel EM64T)		
Red Hat Enterprise Linux(R) 5 Advanced Platform (AMD/Intel 64)		
Red Hat Enterprise Linux(R) 5 (AMD/Intel 64)		
Red Hat Enterprise Linux(R) 6 (64-bit x86_64)		
Red Hat Enterprise Linux(R) 5 Advanced Platform (x86)	Linux 5 (x86)	Linux 5
Red Hat Enterprise Linux(R) 5 (x86)		
Red Hat Enterprise Linux(R) 5 Advanced Platform (AMD/Intel 64)	Linux 5 (AMD/Intel 64)	
Red Hat Enterprise Linux(R) 5 (AMD/Intel 64)		
Red Hat Enterprise Linux(R) 6 (32-bit x86)	Linux 6 (32-bit x86)	Linux 6
Red Hat Enterprise Linux(R) 6 (64-bit x86_64)	Linux 6 (64-bit x86_64)	
turbolinux 7 Server for AP8000	Linux for AP8000	
Microsoft(R) Windows NT(R) Workstation Operating System Version 4.0	Windows NT	
Microsoft(R) Windows NT(R) Server Network Operating System Version 4.0		
Microsoft(R) Windows Server(R) 2003, Standard Edition	Windows Server 2003 Standard Edition	Windows Server 2003
Microsoft(R) Windows Server(R) 2003, Enterprise Edition	Windows Server 2003 Enterprise Edition	
Microsoft(R) Windows Server(R) 2003, Standard x64 Edition	Windows Server 2003 Standard x64 Edition	
Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition	Windows Server 2003 Enterprise x64 Edition	
Microsoft(R) Windows Server(R) 2003 R2, Standard Edition	Windows Server 2003 R2	
Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition		
Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition		
Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition		

Full name or meaning	Abbreviation	
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003 R2, Standard x64 Edition	Windows Server 2003 R2 x64 Editions	
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003 R2, Enterprise x64 Edition		
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 Standard	Windows Server 2008 Standard	Windows Server 2008
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 Enterprise	Windows Server 2008 Enterprise	
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 R2 Standard (x64)	Windows Server 2008 R2	
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 R2 Enterprise (x64)		
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 R2 Datacenter (x64)		
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 Standard (x64)	Windows Server 2008 (x64)	
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2008 Enterprise (x64)		
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2012 Standard	Windows Server 2012	
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2012 Datacenter		
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Standard x64 Edition	Windows Server 2003 x64 Editions	Windows (x64)
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Enterprise x64 Edition		
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003 R2, Standard x64 Edition		
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003 R2, Enterprise x64 Edition		
Microsoft$^{(R)}$ Windows$^{(R)}$ XP Professional x64 Edition	Windows XP x64 Edition	
Microsoft$^{(R)}$ Windows Server$^{(R)}$ 2003, Enterprise x64 Edition	Windows Server 2003 (IPF)	Windows(IPF)
Microsoft$^{(R)}$ Windows$^{(R)}$ XP Professional x64 Edition	Windows XP x64 Edition	Windows XP
Microsoft$^{(R)}$ Windows$^{(R)}$ XP Professional Operating System	Windows XP Professional	
Microsoft$^{(R)}$ Windows$^{(R)}$ XP Home Edition Operating System	Windows XP Home Edition	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Home Basic	Windows Vista Home Basic	Windows Vista
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Home Premium	Windows Vista Home Premium	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Ultimate	Windows Vista Ultimate	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Business	Windows Vista Business	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Enterprise	Windows Vista Enterprise	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Home Basic (x64)	Windows Vista (x64)	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Home Premium (x64)		
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Ultimate (x64)		
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Business (x64)		
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Enterprise (x64)		

Full name or meaning	Abbreviation	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Ultimate (x64)	Windows Vista Ultimate (x64)	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Business (x64)	Windows Vista Business (x64)	
Microsoft$^{(R)}$ Windows Vista$^{(R)}$ Enterprise (x64)	Windows Vista Enterprise (x64)	
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Home Premium	Windows 7 Home Premium	Windows 7
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Professional	Windows 7 Professional	
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Enterprise	Windows 7 Enterprise	
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Ultimate	Windows 7 Ultimate	
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Home Premium (x64)	Windows 7 (x64)	
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Professional (x64)		
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Enterprise (x64)		
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Ultimate (x64)		
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Professional (x64)	Windows 7 Professional (x64)	
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Enterprise (x64)	Windows 7 Enterprise (x64)	
Microsoft$^{(R)}$ Windows$^{(R)}$ 7 Ultimate (x64)	Windows 7 Ultimate (x64)	
Microsoft$^{(R)}$ Windows$^{(R)}$ 8 (Core Edition)	Windows 8 (Core Edition)	Windows 8
Microsoft$^{(R)}$ Windows$^{(R)}$ 8 Pro	Windows 8 Pro	
Microsoft$^{(R)}$ Windows$^{(R)}$ 8 Enterprise	Windows 8 Enterprise	
Microsoft$^{(R)}$ Windows$^{(R)}$ 8 (Core Edition) (x64)	Windows 8 (Core Edition) (x64)	
Microsoft$^{(R)}$ Windows$^{(R)}$ 8 Pro (x64)	Windows 8 Pro (x64)	
Microsoft$^{(R)}$ Windows$^{(R)}$ 8 Enterprise (x64)	Windows 8 Enterprise (x64)	

- Windows Server 2003, Windows Server 2008, and Windows Server 2012 are often referred to collectively as *Windows Server*. Windows XP, Windows Server, Windows Vista, Windows 7, and Windows 8 are often referred to collectively as *Windows*.
- The HiRDB directory path is represented as `%PDDIR%`. The path of the Windows installation directory is represented as `%windir%`.
- The hosts file means the `hosts` file stipulated by TCP/IP. As a rule, a reference to the hosts file means the `%windir%\system32\drivers\etc\hosts` file.

This manual also uses the following abbreviations:

Abbreviation	Full name or meaning
ACK	Acknowledgement
ADM	Adaptable Data Manager

Abbreviation	Full name or meaning
ADO	ActiveX Data Objects
ADT	Abstract Data Type
AP	Application Program
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BES	Back End Server
BLOB	Binary Large Object
BMP	Basic Multilingual Plane
BOM	Byte Order Mark
CD-ROM	Compact Disc - Read Only Memory
CLOB	Character Large Object
CMT	Cassette Magnetic Tape
COBOL	Common Business Oriented Language
CORBA	Common ORB Architecture
CPU	Central Processing Unit
CSV	Comma Separated Values
DAO	Data Access Object
DAT	Digital Audio Tape
DB	Database
DBMS	Database Management System
DDL	Data Definition Language
DIC	Dictionary Server
DLT	Digital Linear Tape
DML	Data Manipulate Language
DNS	Domain Name System
DOM	Document Object Model
DS	Dictionary Server
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DWH	Data Warehouse
EUC	Extended UNIX Code
EX	Exclusive
FAT	File Allocation Table
FD	Floppy Disk
FES	Front End Server
FQDN	Fully Qualified Domain Name

Abbreviation	Full name or meaning
FTP	File Transfer Protocol
GUI	Graphical User Interface
HD	Hard Disk
HDP	Hitachi Dynamic Provisioning
HTML	Hyper Text Markup Language
ID	Identification number
IP	Internet Protocol
IPF	Itanium$^{(R)}$ Processor Family
JAR	Java Archive File
Java VM	Java Virtual Machine
JDBC	Java Database Connectivity
JDK	Java Developer's Kit
JFS	Journaled File System
JFS2	Enhanced Journaled File System
JIS	Japanese Industrial Standard code
JP1	Job Management Partner 1
JRE	Java Runtime Environment
JTA	Java Transaction API
JTS	Java Transaction Service
KEIS	Kanji processing Extended Information System
LAN	Local Area Network
LIP	Loop Initialization Process
LOB	Large Object
LRU	Least Recently Used
LTO	Linear Tape-Open
LU	Logical Unit
LVM	Logical Volume Manager
MGR	System Manager
MIB	Management Information Base
MRCF	Multiple RAID Coupling Feature
MSCS	Microsoft Cluster Server
MSFC	Microsoft Failover Cluster
NAFO	Network Adapter Fail Over
NAPT	Network Address Port Translation
NAT	Network Address Translation
NIC	Network Interface Card

Abbreviation	Full name or meaning
NIS	Network Information Service
NTFS	New Technology File System
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
OLTP	On-Line Transaction Processing
OOCOBOL	Object Oriented COBOL
ORB	Object Request Broker
OS	Operating System
OTS	Object Transaction Service
PC	Personal Computer
PIC	Plug-in Code
PNM	Public Network Management
POSIX	Portable Operating System Interface for UNIX
PP	Program Product
PR	Protected Retrieve
PU	Protected Update
RAID	Redundant Arrays of Inexpensive Disk
RD	Relational Database
RDB	Relational Database
RDO	Remote Data Objects
RiSe	Real time SAN replication
RM	Resource Manager
RMM	Resource Manager Monitor
RPC	Remote Procedure Call
SAX	Simple API for XML
SDS	Single Database Server
SGML	Standard Generalized Markup Language
SJIS	Shift JIS
SNMP	Simple Network Management Protocol
SNTP	Simple Network Time Protocol
SR	Shared Retrieve
SU	Shared Update
TCP/IP	Transmission Control Protocol / Internet Protocol
TM	Transaction Manager
UAP	User Application Program
UOC	User Own Coding

Abbreviation	Full name or meaning
VOS3	Virtual-storage Operating System 3
WS	Workstation
WWW	World Wide Web
XDM/BASE E2	Extensible Data Manager / Base Extended Version 2
XDM/DS	Extensible Data Manager / Data Spreader
XDM/RD E2	Extensible Data Manager / Relational Database Extended Version 2
XDM/XT	Extensible Data Manager / Data Extract
XDS	Extended Data Server
XML	Extensible Markup Language

■ Log representations

The application log that is displayed by Windows Event Viewer is referred to as the *event log*. The following procedure is used to view the event log.

To view the event log:

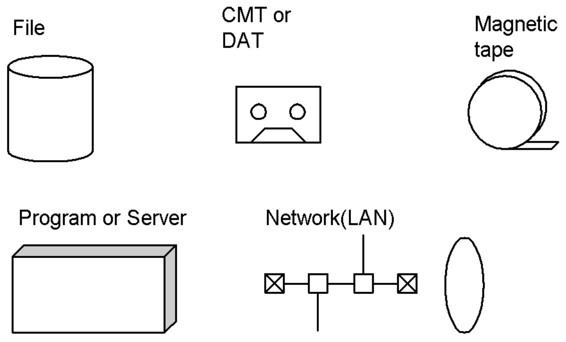
1. Choose **Start**, **Programs**, **Administrative Tools (Common)**, and then **Event Viewer**.
2. Choose **Log**, and then **Application**.

The application log is displayed. Messages with **HiRDBSingleServer** or **HiRDBParallelServer** displayed in the **Source** column were issued by HiRDB.

If you specified a setup identifier when you installed HiRDB, the specified setup identifier follows **HiRDBSingleServer** or **HiRDBParallelServer**.

■ Conventions: Diagrams

This manual uses the following conventions in diagrams:



■ Conventions: Fonts and symbols

The following table explains the text formatting conventions used in this manual:

Text formatting	Convention
Bold	Bold characters indicate text in a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example: • From the **File** menu, choose **Open**. • Click the **Cancel** button.

Text formatting	Convention
	• In the **Enter name** entry box, type your name.
Italic	Italic characters indicate a placeholder for some actual text to be provided by the user or system. For example: • Write the command as follows: `copy` *source-file target-file* • The following message appears: `A file was not found. (file =`*file-name*`)` Italic characters are also used for emphasis. For example: • Do *not* delete the configuration file.
`Monospace`	Monospace characters indicate text that the user enters without change, or text (such as messages) output by the system. For example: • At the prompt, enter `dir`. • Use the `send` command to send mail. • The following message is displayed: `The password is incorrect.`

The following table explains the symbols used in this manual:

Symbol	Convention
|	In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR. For example: `A|B|C` means `A`, or `B`, or `C`.
{ }	In syntax explanations, curly brackets indicate that only one of the enclosed items is to be selected. For example: `{A|B|C}` means only one of `A`, or `B`, or `C`.
[ ]	In syntax explanations, square brackets indicate that the enclosed item or items are optional. For example: `[A]` means that you can specify `A` or nothing. `[B|C]` means that you can specify `B`, or `C`, or nothing.
...	In coding, an ellipsis (`...`) indicates that one or more lines of coding have been omitted. In syntax explanations, an ellipsis indicates that the immediately preceding item can be repeated as many times as necessary. For example: `A, B, B, ...` means that, after you specify `A, B,` you can specify `B` as many times as necessary.
()	Parentheses indicate the range of items to which the vertical bar (|) or ellipsis (`...`) is applicable.
{{ }}	A double pair of curly brackets encloses multiple items, all of which you can specify multiple times as a unit. Example: `{{pdbuffer -a` *option-name*`}}` This example indicates that you can specify the above multiple times as follows: `pdbuffer -a` *option-name* `pdbuffer -a` *option-name*
~	A swung dash precedes the attributes of a user-specified value.
<< >>	A double pair of angle brackets encloses the default value assumed by the system when the specification is omitted.
< >	A single pair of angle brackets encloses the syntax element notation for a user-specified value.
(( ))	A double pair of parentheses encloses the permitted range of values that can be specified.

Symbol	Convention
↑ ↑	Round up the result to the next integer. Example: The result of ↑34 ÷ 3↑ will be 12.
↓ ↓	Discard digits following the decimal point. Example: The result of ↓34 ÷ 3↓ will be 11.
MAX	Select the largest value as the result. Example: The result of MAX(3 × 6, 4 + 7) will be18.
MIN	Select the smallest value as the result. Example: The result of MIN(3 × 6, 4 + 7) will be 11.

Syntax element conventions

The following table explains the syntactical element symbols used in this manual:

Syntactical element symbol	Meaning
<alphabetics>	The alphabetic characters (`A` to `Z` and `a` to `z`) and the underscore (_)
<alphabetics and special characters>	The alphabetic characters (`A` to `Z` and `a` to `z`) and the special characters `#`, `@`, and `\`
<alphanumerics>	Alphabetic characters and the numeric digits (`0` to `9`)
<alphanumerics and special characters>	Alphabetic characters, special characters, and numeric digits
<unsigned integer>	Numeric value
<unsigned decimal>#1	Numeric value (`0` to `9`), period (`.`), numeric value (`0` to `9`)
<hexadecimal>	Numeric digits and `A` to `F` (or `a` to `f`)
<identifier>#2	Alphanumeric character string beginning with an alphabetic character
<symbolic name>	Alphanumeric character string beginning with an alphabetic character or a special character
<character string>	Any string of characters
<path name>#3	Backslash (`\`), alphanumeric characters, period (`.`), space, parentheses, hash mark (`#`), and at mark (`@`)
<host-name>#4	Character string consisting of alphabetic characters (`A` to `Z`, `a` to `z`), numeric characters, period (`.`), hyphen (`-`), underscore (_), and at mark (`@`).

Note: All alphabetic characters must be single-byte characters. Alphabetic characters are case sensitive, but some names might not be case sensitive.

#1

If all the numeric characters preceding the period are zeros (`0`), the zeros preceding the period can be omitted. Similarly, if all the numeric characters following the period are zeros (`0`), the zeros following the period can be omitted.

Example 1: `0.008` ➜ `.008`

Example 2: `15.000` ➜ `15`

#2

Specifying an RDAREA name:

- RDAREA names can include one or more alphanumeric and special characters, underscores (_), hyphens (`-`), and spaces. However, the first character must be an alphabetic or special character.
- When an RDAREA name includes a space, the entire name must be enclosed in double quotation marks (`"`).

#3

Path names depend on the OS being used. If you use a parenthesis or a space in a path name, you must enclose the entire path name in double quotation marks (").

#4

Use one of the following formats to specify a host name:

- *host-name*
- *IP-address* (*xxx*.*xxx*.*xxx*.*xxx* format)
- *fully-qualified-domain-name* (FQDN)

However, do not use different formats to specify the same host. HiRDB regards host names specified in different formats as different hosts.

You can also specify loop-back addresses.

A host name must be registered before it can be specified in a system definition. For details, see *Registering host names* in the *HiRDB Version 9 Installation and Design Guide*.

■ Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is $1{,}024^2$ bytes.
- 1 GB (gigabyte) is $1{,}024^3$ bytes.
- 1 TB (terabyte) is $1{,}024^4$ bytes.

■ Conventions: Version numbers

The version numbers of Hitachi program products are usually written as two sets of two digits each, separated by a hyphen. For example:

- Version 1.00 (or 1.0) is written as 01-00.
- Version 2.05 is written as 02-05.
- Version 2.50 (or 2.5) is written as 02-50.
- Version 12.25 is written as 12-25.

The version number might be shown on the spine of a manual as *Ver. 2.00*, but the same version number would be written in the program as *02-00*.

■ Notes on Windows path names

- In this manual, the Windows terms *directory* and *folder* are both referred to as *directory*.
- Include the drive name when you specify an absolute path name.

Example: `C:\win32app\hitachi\hirdb_s\spool\tmp`

- When you specify in a command argument, control statements file, or HiRDB system definition file a path name that includes a space or a parenthesis, you must enclose the entire path name in double quotation marks (`"`).

Example: `pdinit -d "C:\Program Files(x86)\hitachi\hirdb_s\conf\mkinit"`

However, double quotation marks are not necessary when you use the `set` command in a batch file or at the command prompt to set an environment variable or when you specify the installation directory. If you do use double quotation marks in such a case, the enclosing double quotation marks become part of the value assigned to the environment variable.

Example: `set PDCLTPATH=C:\Program Files\hitachi\hirdb_s\spool`

- HiRDB cannot use files on a networked drive, so you must install HiRDB and configure the HiRDB environment on a local drive. Files used by utilities, such as utility input and output files, must also be on the local drive.

- Do not use short path names (for example, do not use path names such as `C:\PROGRA~1`).

■ Important notes on this manual

The following facilities are explained, but they are not supported:

- Distributed database facility
- Server mode system switchover facility
- User server hot standby
- Rapid system switchover facility
- Standby-less system switchover (1:1) facility
- Standby-less system switchover (effects distributed) facility
- HiRDB External Data Access facility
- Inner replica facility
- Updatable online reorganization
- Sun Java System Directory Server linkage facility
- Simple setup tool
- Extended syslog facility
- Rapid batch facility
- Memory database facility
- Linkage with JP1/NETM/Audit

The following products and option program products are explained, but they are not supported:

- HiRDB CM
- HiRDB Disaster Recovery Light Edition
- uCosminexus Grid Processing Server
- HiRDB Text Search Plug-in
- HiRDB XML Extension
- TP1/Server Base
- JP1/PFM-Agent Option for HiRDB
- JP1/VERITAS NetBackup Agent for HiRDB License
- XDM/RD
- HiRDB SQL Tuning Advisor
- COBOL2002

Contents

1 Overview

This chapter explains the organization of the HiRDB system definitions and the syntax rules for specifying the system definitions.

1.1 For users of a HiRDB/Single Server

The HiRDB administrator creates HiRDB system definitions to set up a HiRDB execution environment. This section explains the following items:

- HiRDB system definition types
- HiRDB system definition creation procedure
- HiRDB system definition file structure
- Relationships among the definitions
- Procedure for modifying the HiRDB system definition (excluding the UAP environment definition)
- Procedure for adding or modifying the UAP environment definition

1.1.1 HiRDB system definition types

The table below lists the types of HiRDB system definitions. The HiRDB administrator must specify values for the operands of the definitions.

Table 1–1: Types of HiRDB system definitions (for a HiRDB/Single Server)

Definition type	Explanation
System common definition	Defines the structure of the HiRDB system and common information. Each HiRDB/Single Server requires a system common definition. For details about the system common definition operands, see Chapter *2. System Common Definition*.
Unit control information definition	Defines control information for a unit. Each unit requires a unit control information definition. For details about the unit control information definition operands, see Chapter *3. Unit Control Information Definition*.
Server common definition	Defines default values for single server definition operands. A server common definition is optional and is defined if needed. For details about the server common definition operands, see Chapter *4. Server Common Definition*.
Single server definition	Defines the execution environment of the single server. Each single server requires a single server definition. For details about the single server definition operands, see Chapter *5. Single Server Definition*.
UAP environment definition	Defines the execution environment of a UAP. UAP environment definitions can be defined as needed. For details about the UAP environment definition operands, see Chapter *9. UAP Environment Definition*. You can create a maximum of 4,096 UAP environment definitions.
SQL reserved word definition	Defines SQL reserved words. Required when using the SQL reserved word deletion facility. For details about the SQL reserved word deletion facility, see *pd_delete_reserved_word_file* operand.

1.1.2 HiRDB system definition creation procedure

The HiRDB administrator uses one of the following tools to create HiRDB system definitions:

- Simple setup tool
- Batch file `SPsetup.bat`
- Text editor such as Notepad

Tip

Normally, you will use the simple setup tool to create the HiRDB system definitions.

Store the created HiRDB system definitions in the files listed in the table below. These files are referred to collectively as *HiRDB system definition files*.

Table 1–2: Storage files for HiRDB system definitions (HiRDB/Single Server)

Definition type	Storage file name
System common definition	`%PDDIR%\conf\pdsys`
Unit control information definition	`%PDDIR%\conf\pdutsys`
Server common definition[#1]	`%PDDIR%\conf\pdsvrc`
Single server definition	`%PDDIR%\conf\`*server-name*[#2]
UAP environment definition	`%PDDIR%\conf\pduapenv\`*any-name*[#3]
SQL reserved word definition	`%PDDIR%\conf\pdrsvwd\`*any-name*[#3]

#1

If a batch file is used to create a HiRDB system definition, the server common definition is not created. If a server common definition is needed, we recommend that the HiRDB administrator use a text editor such as Notepad to create it.

#2

This must be the server name specified in the `-s` option of the `pdstart` operand in the system common definition.

If the simple setup tool or a batch file is used to create the HiRDB system definition, the file name is `sds01`.

#3

- A file name is an alphanumeric character string (a maximum of eight characters) that begins with a letter. The file name is not case sensitive. That is, `A` and `a` are considered the same, and file names `ABC` and `abc` are the same.
- Make sure that the all users who will use the UAP environment definition or SQL reserved word definition are assigned read privilege (`r`) and execution privilege (`x`) for the directory in which each file is stored. Also make sure that each file is assigned read privilege (`r`).

! Important note

Set and maintain the permissions for the HiRDB system definition files so that only the file owner (HiRDB administrator) has read and write privileges.

(1) Using the simple setup tool

When the simple setup tool is used to create HiRDB system definitions, the simple setup tool itself sets up the HiRDB environment. The HiRDB system definitions are created automatically based on information specified by using the simple setup tool and are stored in the files listed in *Table 1-2 Storage files for HiRDB system definitions (HiRDB/ Single Server)*. If necessary, the HiRDB administrator can use the simple setup tool to modify the definition contents (the values specified for the operands).

Note that a UAP environment definition or an SQL reserved word definition cannot be created using the simple setup tool.

(2) Using the batch file SPsetup.bat

When the batch file is used to set up a HiRDB environment, HiRDB automatically creates HiRDB system definitions (which are stored in the files shown in Table 1-2). If necessary, the HiRDB administrator can use a text editor such as Notepad to modify the definition contents (operand specification values).

Note that a UAP environment definition or an SQL reserved word definition cannot be created in a batch file.

(3) Using a text editor such as Notepad

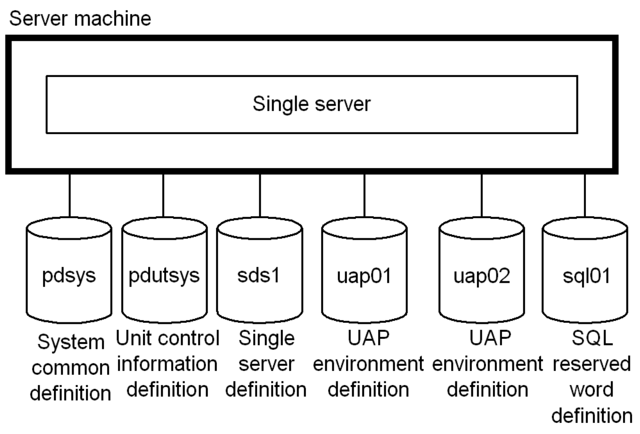
To set up a HiRDB environment using commands, use a text editor such as Notepad to create HiRDB system definitions. Specify each HiRDB system definition operand as appropriate to the HiRDB execution environment.

Once the HiRDB system definitions have been created, use the `pdconfchk` command to check the integrity of the HiRDB system definition operands; for details about the operands that are checked, see *E. Operands Checked by the pdconfchk Command*.

1.1.3 HiRDB system definition file structure

When the simple setup tool or the batch file is used to create the HiRDB system definitions, HiRDB creates the HiRDB system definition files as shown in Figure 1-1. We recommend that you also reference Figure 1-1 if you use a text editor such as Notepad to create the HiRDB system definitions.

Figure 1–1: Structure of HiRDB system definition files (HiRDB/Single Server)


Server machine
Single server
pdsys
pdutsys
sds1
uap01
uap02
sql01
System common definition
Unit control information definition
Single server definition
UAP environment definition
UAP environment definition
SQL reserved word definition

1.1.4 Relationships among HiRDB system definitions

If the same operand is defined in different definitions, HiRDB determines the value to be used on the basis of the priority order of the definitions, as shown below:

1. Single server definition
2. Server common definition
3. Unit control information definition
4. System common definition

A standard value can be specified in a definition that has a lower priority, and then that value can be changed by specifying a different value in a definition that has a higher priority.

(1) Relationship to client environment definition

The values of the HiRDB system definition operands shown in the following table can be modified for each client in its client environment definition.

Table 1–3: Operands with specification values that can be modified in client environment definitions (HiRDB/Single Server)

HiRDB system definition operand	Client environment definition operand	Operand contents
`pd_additional_optimize_level`	`PDADDITIONALOPTLVL`	Specifies the SQL extension optimizing option.
`pd_cwaittime_wrn_pnt`	`PDCWAITTIMEWRNPNT`	Specifies the condition for outputting the SQL runtime warning information of the SQL runtime warning output facility as a percentage of the client's maximum

HiRDB system definition operand	Client environment definition operand	Operand contents
		wait time (value specified by the `PDCWAITTIME` operand in the client environment definition).
`pd_delete_reserved_word_file`	`PDDELRSVWDFILE`	Specifies the name of the SQL reserved word deletion file when using the SQL reserved word deletion facility.
`pd_hash_table_size`	`PDHASHTBLSIZE`	Specifies the hash table size when hash execution of a hash join or a subquery is used as the SQL extension optimizing option.
`pd_hashjoin_hashing_mode`	`PDHJHASHINGMODE`	Specifies the hashing method when hash join, subquery hash execution is specified as the SQL extension optimizing option.
`pd_optimize_level`	`PDSQLOPTLVL`	Specifies the SQL optimization option.
`pd_space_level`	`PDSPACELVL`	Specifies a space conversion level.
`pd_uap_exerror_log_param_size`	`PDUAPEXERLOGPRMSZ`	Specifies the maximum data size of the parameter information to be output to the client error log file or to the SQL error report file.
`pd_uap_exerror_log_use`	`PDUAPEXERLOGUSE`	Specifies whether to use the facility for output of extended SQL error information.
`pd_watch_pc_client_time`	`PDSWATCHTIME`	Specifies the amount of time to wait for a request from a Windows-compatible HiRDB client.

To change the value for a client, specify the applicable operand in the client's environment definition. For details about client environment definition, see the *HiRDB Version 9 UAP Development Guide*.

1.1.5 Procedure for modifying the HiRDB system definitions (excluding the UAP environment definition)

(1) Modifying the HiRDB system definitions

This subsection describes how to modify the HiRDB system definitions. Note that `%PDDIR%\conf` is the directory that stores the unit control information definition file. `%PDCONFPATH%` is the directory that stores other HiRDB system definition files.

To modify the HiRDB system definitions:

1. Create subdirectories under `%PDDIR%\conf` and `%PDCONFPATH%`. In this example, a subdirectory called `work` is created.
2. Copy the unit control information definition file to `%PDDIR%\conf\work`. Copy other HiRDB system definition files to `%PDCONFPATH%\work`.
3. Modify the HiRDB system definitions that have been copied to `%PDDIR%\conf\work` and `%PDCONFPATH%\work`.
4. Use the `pdconfchk -d work` command to check the content of the HiRDB system definitions in `%PDDIR%\conf\work` and `%PDCONFPATH%\work`. If any error is found, correct the HiRDB system definition and re-execute the `pdconfchk` command.

5. Use the `pdstop` command to normally terminate HiRDB.
6. Use the `pdlogunld` command to unload the system log files that are waiting to be unloaded.
7. Replace the HiRDB system definition file by copying the HiRDB system definition file modified in step 3 to `%PDDIR%\conf` or `%PDCONFPATH%`.
8. If you modify the specified values of the following operands, you must use the `pdloginit` command to initialize the system log file.
   - `pd_log_dual`
   - `pdstart`
9. Use the `pdstart` command to normally start the HiRDB.

(2) Using the system reconfiguration command to modify the HiRDB system definitions

If you use the system reconfiguration command (`pdchgconf` command), you can modify the HiRDB system definitions while HiRDB is running, and therefore you need not normally terminate HiRDB. Note however that HiRDB Advanced High Availability is required for using this command.

To modify the HiRDB system definition using the system reconfiguration command:

1. Create the `%PDDIR%\conf\chgconf` directory.
2. Copy the HiRDB system definition file being used to the directory created in step 1.
3. Modify the HiRDB system definitions in `%PDDIR%\conf\chgconf`.
4. Use the `pdconfchk -d chgconf` command to check the HiRDB system definitions in `%PDDIR%\conf\chgconf`. If any error is found, correct the HiRDB system definition and re-execute the `pdconfchk` command.
5. Use the `pdchgconf` command to replace the HiRDB system definition with the modified one.
   When you execute the `pdchgconf` command, the HiRDB system definition file being used (unmodified one) is saved to `%PDDIR%\conf\backconf`. Then, the modified HiRDB system definition file in `%PDDIR%\conf\chgconf` is copied to `%PDDIR%\conf`.

Notes:

- If a transaction or utility continues to be active for 15 minutes or longer after the `pdchgconf` command is input, the command is terminated abnormally.
- Some limits apply when you use the system reconfiguration command to modify the HiRDB system definitions. For details about these limits, see the *HiRDB Version 9 System Operation Guide*.

(3) Notes

- HiRDB system definitions must not be modified or deleted while they are being used by an active HiRDB. If such a definition is modified or deleted, the operation of the HiRDB cannot be guaranteed.
- Once a planned, forced, or abnormal termination of HiRDB occurs, some of the HiRDB system definition operands cannot be modified; for details, see *1.4 List of operands* or the operand explanations for the various definition statements.
- After you have modified the HiRDB system definitions, back up the files located in `%PDDIR%\conf`. To guard against errors in the disk containing the HiRDB directory, back up the files in the HiRDB directory (files in `%PDDIR%\conf`). To restore the HiRDB directory, you need the backup of the files in `%PDDIR%\conf`. Also back up`%PDCONFPATH%` if it is located under the HiRDB directory.

(4) Linking a HiRDB Datareplicator

Before the operands listed below are added, modified, or deleted, HiRDB Datareplicator must be terminated; HiRDB Datareplicator can be restarted after the operand addition, modification, or deletion operation is completed:

- `pd_log_dual`
- `pd_log_max_data_size`

- `pdlogadfg -d sys`
- `pdlogadpf -d sys`

If these operands are added, modified, or deleted while HiRDB Datareplicator is running, extraction by HiRDB Datareplicator might fail under some circumstances.

1.1.6 Adding or modifying the UAP environment definition

To add or modify the UAP environment definition:

1. Make sure that the UAP that uses the UAP environment definition to be modified is not active. If it is active, wait until it is terminated. If a UAP environment definition is added or modified while the UAP is active, the unmodified UAP environment definition is applied to the active UAP. However, depending on the timing, the modified UAP environment definition might be applied to the UAP.
2. Add or modify the UAP environment definition.
3. Use the modified UAP environment definition to execute the UAP.

1.2 For users of a HiRDB/Parallel Server

The HiRDB administrator creates HiRDB system definitions to set up a HiRDB execution environment. This section explains the following items:

- HiRDB system definition types
- HiRDB system definition creation procedure
- HiRDB system definition file structure
- Relationships among the definitions
- Procedure for modifying the HiRDB system definition (excluding the UAP environment definition)
- Adding or modifying the UAP environment definition
- Procedure for creating a HiRDB system definition when using the standby-less system switchover (effects distributed) facility

1.2.1 HiRDB system definition types

The table below lists the types of HiRDB system definitions. The HiRDB administrator must specify values for the operands of the definitions.

Table 1–4: Types of HiRDB system definitions (for a HiRDB/Parallel Server)

Definition type		Explanation
System common definition		Defines the structure of the HiRDB system and common information. Each unit requires a system common definition. *The contents of the individual system common definitions must be identical for all units.* For details about the system common definition operands, see Chapter *2. System Common Definition*.
Unit control information definition		Defines control information for a unit. Each unit requires a unit control information definition. For details about the unit control information definition operands, see Chapter *3. Unit Control Information Definition*.
Server definitions	Server common definition	Defines default values for server definition operands. A server common definition is optional and is defined if needed. For details about the server common definition operands, see Chapter *4. Server Common Definition*.
	Front-end server definition	Defines the execution environment of a front-end server. Each front-end server requires a front-end server definition. For details about the front-end server definition operands, see Chapter *6. Front-End Server Definition*.
	Dictionary server definition	Defines the execution environment of a dictionary server. Each dictionary server requires a dictionary server definition. For details about the dictionary server definition operands, see Chapter *7. Dictionary Server Definition*.
	Back-end server definition	Defines the execution environment of a back-end server. Each back-end server requires a back-end server definition. For details about the back-end server definition operands, see Chapter *8. Back-End Server Definition*.
UAP environment definition		Defines the execution environment of a UAP. UAP environment definitions can be defined as needed. Create the UAP environment definition in the unit in which a front-end server is located. If there are multiple front-end servers, create the definition for the front-end server to which the UAP environment definition is to be applied. For details about the UAP environment definition operands, see Chapter *9. UAP Environment Definition*. You can create a maximum of 4,096 UAP environment definitions.
SQL reserved word definition		Required when using the SQL reserved word deletion facility. Defines SQL reserved words. Create the SQL reserved word definition in the unit in which a front-end server is located. If there are multiple front-end servers, you must create the same SQL reserved word definition for all units in which the front-end servers are located. For details about the SQL reserved word deletion facility, see *pd_delete_reserved_word_file* operand.

1.2.2 HiRDB system definition creation procedure

The HiRDB administrator uses one of the following tools to create HiRDB system definitions:

- Simple setup tool
- Text editor such as Notepad

Tip

Normally, you will use the simple setup tool to create the HiRDB system definitions.

The created HiRDB system definitions must be stored in the files listed in the table below. These files are referred to collectively as *HiRDB system definition files*.

Table 1–5: Storage files for HiRDB system definitions (HiRDB/Parallel Server)

Definition type	Storage file name
System common definition	`%PDDIR%\conf\pdsys`
Unit control information definition	`%PDDIR%\conf\pdutsys`
Server common definition	`%PDDIR%\conf\pdsvrc`
Front-end server definition	`%PDDIR%\conf\`*server-name*#1
Dictionary server definition	`%PDDIR%\conf\`*server-name*#1
Back-end server definition	`%PDDIR%\conf\`*server-name*#1
UAP environment definition	`%PDDIR%\conf\pduapenv\`*any-name*#2
SQL reserved word definition	`%PDDIR%\conf\pdrsvwd\`*any-name*#2

#1

This must be the server name specified in the `-s` option of the `pdstart` operand in the system common definition.

#2

- A file name is an alphanumeric character string (a maximum of eight characters) that begins with a letter. The file name is not case sensitive. That is, `A` and `a` are considered the same, and file names `ABC` and `abc` are the same.
- Make sure that all users who will use the UAP environment definition or SQL reserved word definition are assigned read privilege (`r`) and execution privilege (`x`) for the directory in which each file is stored. Also make sure that each file is assigned read privilege (`r`).

Important note

Set and maintain the permissions for the HiRDB system definition files so that only the file owner (HiRDB administrator) has read and write privileges.

(1) Using the simple setup tool

When the simple setup tool is used to create HiRDB system definitions, the simple setup tool itself sets up the HiRDB environment. The HiRDB system definitions are created automatically based on information specified by using the simple setup tool and are stored in the files listed in *Table 1-5 Storage files for HiRDB system definitions (HiRDB/Parallel Server)*. If necessary, the HiRDB administrator can use the simple setup tool to modify the definition contents (the values specified for the operands).

Note that the following definitions cannot be created using the simple setup tool:

- UAP environment definition
- SQL reserved word definition

(2) Using a text editor such as Notepad

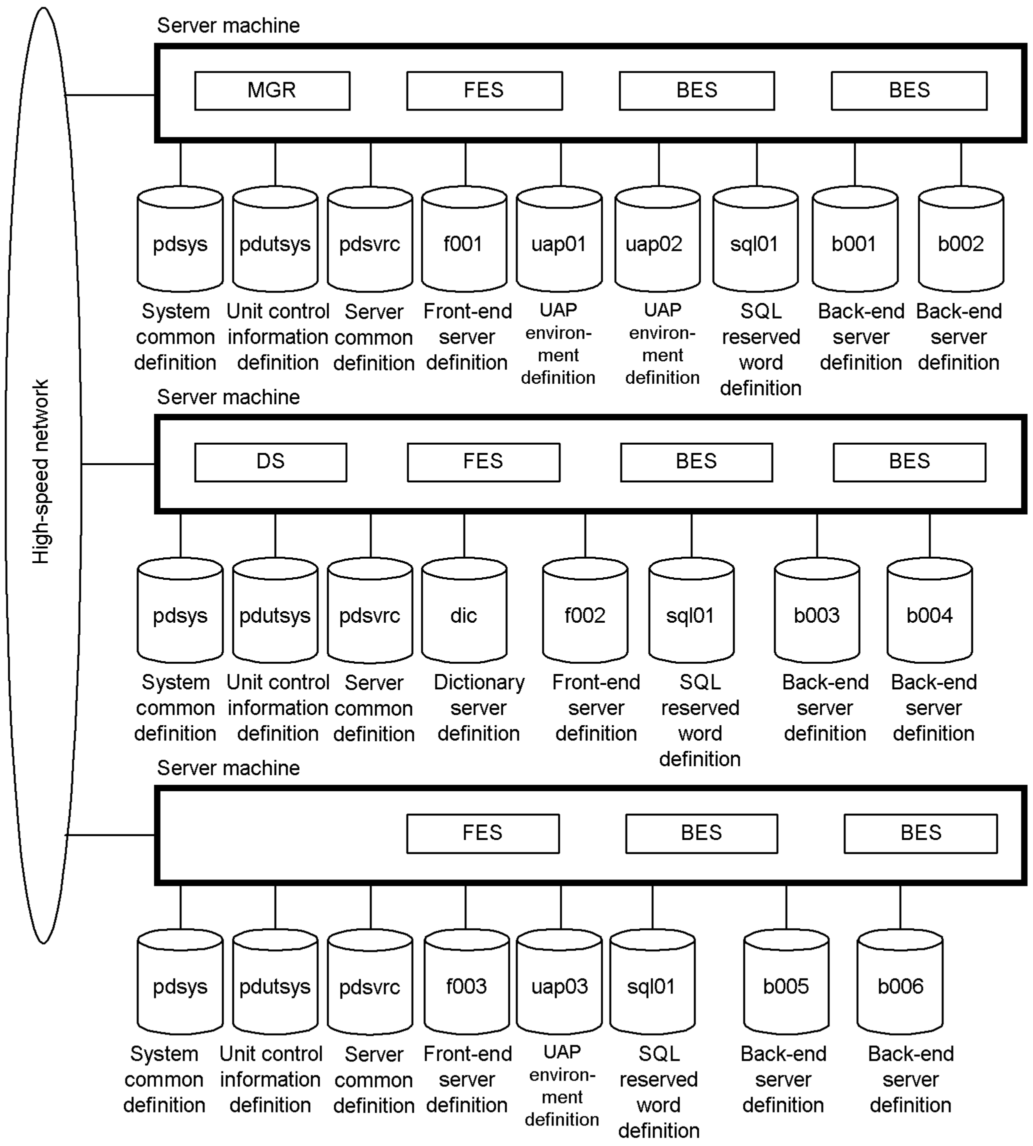
To set up a HiRDB environment using commands, use a text editor such as Notepad to create HiRDB system definitions. Specify each HiRDB system definition operand as appropriate for the HiRDB execution environment.

Once the HiRDB system definitions have been created, use the `pdconfchk` command to check the integrity of the HiRDB system definition operands. For details about the operands that are checked, see *E. Operands Checked by the pdconfchk Command*.

1.2.3 HiRDB system definition file structure

When the simple setup tool is used to create the HiRDB system definitions, HiRDB creates the HiRDB system definition files as shown in Figure 1-2. We recommend that you also reference Figure 1-2 if you use a text editor such as Notepad to create the HiRDB system definitions.

Figure 1–2: Structure of HiRDB system definition files (HiRDB/Parallel Server)


High-speed network
Server machine
MGR
FES
BES
BES
pdsys
pdutsys
pdsvrc
f001
uap01
uap02
sql01
b001
b002
System common definition
Unit control information definition
Server common definition
Front-end server definition
UAP environ-ment definition
UAP environ-ment definition
SQL reserved word definition
Back-end server definition
Back-end server definition
Server machine
DS
FES
BES
BES
pdsys
pdutsys
pdsvrc
dic
f002
sql01
b003
b004
System common definition
Unit control information definition
Server common definition
Dictionary server definition
Front-end server definition
SQL reserved word definition
Back-end server definition
Back-end server definition
Server machine
FES
BES
BES
pdsys
pdutsys
pdsvrc
f003
uap03
sql01
b005
b006
System common definition
Unit control information definition
Server common definition
Front-end server definition
UAP environ-ment definition
SQL reserved word definition
Back-end server definition
Back-end server definition

Note 1: The system common definition must be the same for all server machines.
Note 2: The SQL reserved word definition must be the same for all server machines.

1.2.4 Relationships among HiRDB system definitions

If the same operand is defined in different definitions, HiRDB determines the value to be used on the basis of the priority order of the following definitions:

1. Front-end server definition, dictionary server definition, or back-end server definition
2. Server common definition
3. Unit control information definition
4. System common definition

A standard value can be specified in a definition of a lower priority, and then that value can be changed by specifying a different value in a definition of a higher priority. For example, a standard value can be specified in the system common definition and then this value can be changed by specifying values in the server definitions of individual servers.

(1) Relationship to client environment definition

The values of the HiRDB system definition operands listed in the following table can be modified for each client in its client environment definition.

Table 1–6: Operands with specification values that can be modified in client environment definitions (HiRDB/Parallel Server)

HiRDB system definition operand	Client environment definition operand	Operand contents
pd_additional_optimize_level	PDADDITIONALOPTLVL	Specifies the SQL extension optimizing option.
pd_cwaittime_wrn_pnt	PDCWAITTIMEWRNPNT	Specifies the condition for outputting the SQL runtime warning information of the SQL runtime warning output facility as a percentage of the client's maximum wait time (value specified by the PDCWAITTIME operand in the client environment definition).
pd_delete_reserved_word_file	PDDELRSVWDFILE	Specifies the name of the SQL reserved word deletion file when using the SQL reserved word deletion facility.
pd_ha_transaction	PDHATRNQUEUING	Specifies whether to use the transaction queuing facility.
pd_hash_table_size	PDHASHTBLSIZE	Specifies the hash table size when hash execution of a hash join or a subquery is used as the SQL extension optimizing option.
pd_hashjoin_hashing_mode	PDHJHASHINGMODE	Specifies the hashing method when hash join, subquery hash execution is specified as the SQL extension optimizing option.
pd_optimize_level	PDSQLOPTLVL	Specifies the SQL optimization option.
pd_space_level	PDSPACELVL	Specifies a space conversion level.
pd_uap_exerror_log_param_size	PDUAPEXERLOGPRMSZ	Specifies the maximum data size of the parameter information to be output to the client error log file or to the SQL error report file.

HiRDB system definition operand	Client environment definition operand	Operand contents
`pd_uap_exerror_log_use`	`PDUAPEXERLOGUSE`	Specifies whether to use the facility for output of extended SQL error information.
`pd_watch_pc_client_time`	`PDSWATCHTIME`	Specifies the amount of time to wait for a request from a Windows-compatible HiRDB client.

To change the value for a client, the applicable operand is specified in the client's environment definition. For details about client environment definition, see the *HiRDB Version 9 UAP Development Guide*.

1.2.5 Procedure for modifying the HiRDB system definitions (excluding UAP environment definition)

(1) Modifying the HiRDB system definitions

This subsection describes how to modify the HiRDB system definitions. Note that `%PDDIR%\conf` is the directory that stores the unit control information definition file. `%PDCONFPATH%` is the directory that stores other HiRDB system definition files.

To modify the HiRDB system definitions:

1. Create subdirectories under `%PDDIR%\conf` and `%PDCONFPATH%`. In this example, a subdirectory called `work` is created.
2. Copy the unit control information definition file to `%PDDIR%\conf\work`. Copy other HiRDB system definition files to `%PDCONFPATH%\work`.
3. Modify the HiRDB system definitions that have been copied to `%PDDIR%\conf\work` and `%PDCONFPATH%\work`.
4. Use the `pdconfchk -d work` command to check the content of the HiRDB system definitions in `%PDDIR%\conf\work` and `%PDCONFPATH%\work`. If any error is found, correct the HiRDB system definition and re-execute the `pdconfchk` command.
5. Use the `pdstop` command to normally terminate HiRDB.
6. Use the `pdlogunld` command to unload the system log files that are waiting to be unloaded.
7. Replace the HiRDB system definition file by copying the HiRDB system definition file modified in step 3 to `%PDDIR%\conf` or `%PDCONFPATH%`.
8. If you modify the specified values of the following operands, you must use the `pdloginit` command to initialize the system log file.
   - `pd_log_dual`
   - `pdstart`
9. Use the `pdstart` command to normally start HiRDB.

(2) Using the system reconfiguration command to modify the HiRDB system definitions

If you use the system reconfiguration command (`pdchgconf` command), you can modify the HiRDB system definitions while HiRDB is running, and therefore you need not normally terminate HiRDB. Note however that HiRDB Advanced High Availability is required for using this command.

To modify the HiRDB system definition using the system reconfiguration command:

1. Create the `%PDDIR%\conf\chgconf` directory.
2. Copy the HiRDB system definition file being used to the directory created in step 1.
3. Modify the HiRDB system definitions in `%PDDIR%\conf\chgconf`.

4. Use the `pdconfchk -d chgconf` command to check the HiRDB system definitions in `%PDDIR%\conf\chgconf`. If any error is found, correct the HiRDB system definition and re-execute the `pdconfchk` command.
5. Use the `pdchgconf` command to replace the HiRDB system definition with the modified one.

   When you execute the `pdchgconf` command, the HiRDB system definition file being used (unmodified one) is saved to `%PDDIR%\conf\backconf`. Then, the modified HiRDB system definition file in `%PDDIR%\conf\chgconf` is copied to `%PDDIR%\conf`.

Notes:

- If a transaction or utility continues to be active for 15 minutes or longer after the `pdchgconf` command is input, the command is aborted.
- Some limits apply when you use the system reconfiguration command to modify the HiRDB system definitions. For details about these limits, see the *HiRDB Version 9 System Operation Guide*.

(3) Notes

- When a modification is made to a system common definition, the identical modification must be made to the system common definitions at all server machines.
- If some of the units are abnormally terminated during normal or planned termination of HiRDB, do not modify the HiRDB system definition before restarting HiRDB. If it is modified, HiRDB will not start. Even if HiRDB starts, it cannot run normally after the startup.
- For a HiRDB/Parallel Server, create subdirectories under `%PDDIR%\conf` and `%PDCONFPATH%` for each unit, and check the content of the HiRDB system definitions.
- HiRDB system definitions must not be modified or deleted while they are being used by an active HiRDB. If such a definition is modified or deleted, the operation of the HiRDB cannot be guaranteed.
- Once a planned, forced, or abnormal termination of HiRDB occurs, some of the HiRDB system definition operands cannot be modified; for details, see *1.4 List of operands* or the operand explanations for the various definition statements.
- After you have modified the HiRDB system definitions, back up the files located in `%PDDIR%\conf`. To guard against errors in the disk containing the HiRDB directory, back up the files in the HiRDB directory (files in `%PDDIR%\conf`). To restore the HiRDB directory, you need the backup of the files in `%PDDIR%\conf`. Also back up `%PDCONFPATH%` if it is located under the HiRDB directory.
- Note the following when using the standby-less system switchover (1:1) facility: Before modifying the HiRDB system definitions of a normal BES unit, first use the `pdstop -u` command to normally terminate the normal BES unit and the alternate BES unit. After modifying the HiRDB system definitions, copy both the unit control information definition file and back-end server definition file of the normal BES unit to the alternate BES unit. For details, see the section *Creating HiRDB system definitions (system switchover)* in the *HiRDB Version 9 System Operation Guide*.

(4) When linking a HiRDB Datareplicator

Before the operands listed below are added, modified, or deleted, HiRDB Datareplicator must be terminated; HiRDB Datareplicator can be restarted after the operand addition, modification, or deletion operation is completed:

- `pd_log_dual`
- `pd_log_max_data_size`
- `pdlogadfg -d sys`
- `pdlogadpf -d sys`

If these operands are added, modified, or deleted while HiRDB Datareplicator is running, extraction by HiRDB Datareplicator might fail under some circumstances.

1.2.6 Adding or modifying the UAP environment definition

To add or modify the UAP environment definition:

1. Make sure that the UAP that uses the UAP environment definition to be modified is not active. If it is active, wait until it is terminated. If a UAP environment definition is added or modified while the UAP is active, the unmodified UAP environment definition is applied to the active UAP. However, depending on the timing, the modified UAP environment definition might be applied to the UAP.
2. Add or modify the UAP environment definition.
3. Use the modified UAP environment definition to execute the UAP.

1.2.7 Creating a HiRDB system definition when using the standby-less system switchover (effects distributed) facility

This section explains how to create a HiRDB system definition when using the standby-less system switchover (effects distributed) facility and provides related notes.

Key point

The operating environment of the back-end server for the unit in which the host BES is running must match the operating environment of the back-end server for the unit in which the guest BES is running.

(1) System common definition

Use the same system common definition for all units. Copy the system common definition that has been created.

Note:

Specify the operand that is specified as the default value of the back-end server definition in the system common definition. An example of such an operand is the `pd_sql_object_cache_size` operand. You cannot specify this operand in the server common definition only.

(2) Unit control information definition

Some restrictions apply to the operands that can be specified in the unit control information definition of the unit to which the standby-less system switchover (effects distributed) facility is to be applied. For details about the operands that can be specified in the unit control information definition, see *F. List of Operands That Can Be Specified When Using the Standby-less System Switchover (Effects Distributed) Facility (Unit Control Information Definition)*.

If you must specify an operand that cannot be specified in the unit control information definition, specify it in the system common definition or back-end server definition.

Notes:

If an operand satisfies either of the following conditions, it must be specified in the same way in all of the unit control information definitions of the units that comprise an HA group.

- Operand that can be specified in both the unit control information definition and back-end server definition
- Operand in the unit control information definition that affects the operation of the back-end server

(3) Server common definition

Units that comprise an HA group use the same server common definition. Copy the server common definition that has been created.

(4) Back-end server definition

Define the back-end server definition of the host BES in each of the units that comprise an HA group. Copy the back-end server definition that has been created.

1.3 Coding format of HiRDB system definitions

This section explains the syntax rules for coding the HiRDB system definition operands.

(1) Operand specification format

HiRDB system definition operands are specified in the following three formats:

set format

`set` is used to set a value in an operand; for example,

```
set pd_max_users=15
```

Command format

Options and command arguments are set in an operand; for example,

```
pdlogadfg -d sys -g loggrp01 ONL
```

`pdlogadfg`

Command name

`-d sys -g loggrp01`

Options (an option is a character string that begins with a hyphen)

Flag arguments are not specified in Format 1; flag arguments are specified in Format 2:

- Format 1: *option-flag*
- Format 2: *option-flag flag-argument*

  option-flag: An alphabetic character that follows the hyphen.

  Values are case sensitive.

  flag-argument: Operation target for the option flag.

`ONL`

A command argument (argument that begins with anything other than a hyphen).

putenv format

`putenv` is used to set in an operand an environment variable and value for the environment variable; for example,

```
putenv SHMMAX 16
```

(2) Comments

A comment can be entered for any operand. A comment must begin with the number sign (#). When the number sign is encountered, the remainder of the line is assumed to be a comment. When the number sign is entered at the beginning of a line, the entire line is handled as a comment line.

Example

```
set pd_max_users=15  # maximum number of concurrent connections
pdlogadfg -d sys -g loggrp01 ONL
# Defines a file group for system log files
```

(3) Line continuation

The maximum number of characters per line of definition is 80. If a definition requires more than 80 characters, additional lines can be used by specifying the backslash (\) as a continuation symbol before the end of each line that is to be continued.

Example

```
pdbuffer -a buffer ABC -n 160 -r rdareaA,rdareaB,... \
rdareaZ
```

A line in which a comment is specified cannot be continued. When the hash mark (#) is encountered, the entire remainder of the line is handled as a comment line, so even if the backslash is specified subsequently on the line, it will be regarded as part of the comment rather than as the line continuation symbol.

(4) Duplicate operand specification

When multiple operands are specified in a single definition statement (single file), HiRDB uses the following rules for processing these operands:

- When the operands are in the `set` or `putenv` format, the value of the operand that was specified last takes precedence.
- When the operands are in the command format, priority depends on the operands. For details, see the descriptions of the individual operands.

(5) Notes

- To change a value specified with the `putenv` format operand in the HiRDB system definition to its default value, set the default value (do not delete the operand).
- Do not use a file extension such as `.txt` for a HiRDB system definition file.
- When specifying an absolute path name, be sure to include the drive name.
  Example: `C:\hirdb_s\spool\tmp`
- Enclose in double quotation marks (") any path name specified in the HiRDB system definition file that includes a space.
  Example:
  `pdinit -d "C:\Program Files\hitachi\hirdb_s\conf\mkinit"`

1.4 List of operands

1.4.1 Operands with modifiable specifications

The table below lists operands defined in the HiRDB system definitions and whether each operand can be modified when HiRDB is restarted. The operands are listed in alphabetical order. Note that the operands of the definition in the bulleted line below are not described here; those operands can be modified before the HiRDB system is restarted (after a planned, forced, or abnormal termination).

- UAP environment definition

Table 1–7: List of operands defined in the HiRDB system definitions and whether they can be modified when HiRDB is restarted

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_additional_optimize_level[#6]	Y	--	--	--	Y	--	--	Y	Y
pd_alv_port	Y	--	--	--	--	--	--	Y	Y
pd_apply_search_ats_num	Y	--	--	--	--	--	--	Y	Y
pd_assurance_index_no	Y	--	--	--	--	--	--	Y[#4]	Y
pd_assurance_table_no	Y	--	--	--	--	--	--	Y[#4]	Y
pd_audit	Y	Y	--	--	--	--	--	N	N
pd_aud_async_buff_count	Y	Y	--	--	--	--	--	Y	Y
pd_aud_async_buff_retry_intvl	Y	Y	--	--	--	--	--	Y	Y
pd_aud_async_buff_size	Y	Y	--	--	--	--	--	Y	Y
pd_aud_auto_loading	Y	--	--	--	--	--	--	N	N
pd_aud_file_name	Y	Y	--	--	--	--	--	Y	Y
pd_aud_file_wrn_pnt	Y	--	--	--	--	--	--	Y	Y
pd_aud_max_generation_num	Y	Y	--	--	--	--	--	Y	Y
pd_aud_max_generation_size	Y	Y	--	--	--	--	--	Y	Y
pd_aud_no_standby_file_opr	Y	--	--	--	--	--	--	Y	Y
pd_aud_sql_data_size	Y	Y	--	--	--	--	--	Y	Y
pd_aud_sql_source_size	Y	Y	--	--	--	--	--	Y	Y
pd_audit_def_buffer_size	--	--	Y	Y	Y	--	--	Y	Y
pd_auth_cache_size	--	--	Y	Y	Y	--	--	Y[#2, #3]	Y[#2]
pd_auto_vrup	Y	--	--	--	--	--	--	N	N

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_bes_connection_hold	--	--	Y	--	--	--	Y	Y	Y
pd_bes_conn_hold_trn_interval	--	--	Y	--	--	--	Y	Y	Y
pd_bes_shmpool_size	--	--	Y	--	--	--	Y	Y#2	Y#2
pd_c_library_directory	Y	Y	--	--	--	--	--	Y	Y
pd_cancel_down_msgchange	Y	--	--	--	--	--	--	Y	Y
pd_cancel_dump	Y	Y	--	--	--	--	--	Y	Y
pd_change_clt_ipaddr	Y	Y	--	--	--	--	--	Y	Y
pd_check_pending	Y	--	--	--	--	--	--	Y	Y
pd_client_waittime_over_abort	Y	--	--	--	--	--	--	Y	Y
pd_clt_waittime_over_dump_level	Y	--	--	--	--	--	--	Y	Y
pd_cmdhold_precheck	Y	--	--	--	--	--	--	Y	Y
pd_command_deadlock_priority	Y	--	--	--	--	--	--	Y	Y
pd_connect_errmsg_hide	Y	--	--	--	--	--	--	Y	Y
pd_constraint_name	Y	--	--	--	--	--	--	Y	Y
pd_cwaittime_report_dir	Y	Y	--	--	--	--	--	Y	Y
pd_cwaittime_report_size	Y	Y	--	--	--	--	--	Y	Y
pd_cwaittime_wrn_pnt#6	Y	--	--	Y	Y	--	--	Y	Y
pd_db_access_error_action	Y	--	--	--	--	--	--	Y	Y
pd_db_hold_action	Y	--	--	--	--	--	--	Y	Y
pd_dbbuff_attribute	Y	Y	--	--	--	--	--	Y	Y
pd_dbbuff_binary_data_lru	Y	--	--	--	--	--	--	Y	Y
pd_dbbuff_lock_interval	Y	--	--	--	--	--	--	Y	Y
pd_dbbuff_lock_release_detect	Y	--	--	--	--	--	--	Y	Y
pd_dbbuff_lock_spn_count	Y	--	--	--	--	--	--	Y	Y
pd_dbbuff_lru_option	Y	--	--	--	--	--	--	Y	Y
pd_dbbuff_modify	Y	--	--	--	--	--	--	N	N
pd_dbbuff_rate_updpage	Y	--	--	--	--	--	--	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
`pd_dbbuff_trace_level`	Y	--	--	--	--	--	--	Y	Y
`pd_dbbuff_wait_interval`	Y	Y	--	--	--	--	--	Y	Y
`pd_dbbuff_wait_spn_count`	Y	Y	--	--	--	--	--	Y	Y
`pd_dbreuse_remaining_entries`	Y	--	--	--	--	--	--	Y	Y
`pd_dbsync_altwrite_skip`	Y	--	--	--	--	--	--	Y	Y
`pd_dbsync_lck_release_count`	--	--	Y	Y	--	Y	Y	Y	Y
`pd_dbsync_point`	Y	--	--	--	--	--	--	N	Y
`pd_db_io_error_action`	Y	Y	--	--	--	--	--	Y	Y
`pd_deadlock_priority_use`	Y	--	--	--	--	--	--	Y	Y
`pd_debug_info_netstat`	Y	--	--	--	--	--	--	Y	Y
`pd_dec_sign_normalize`	Y	--	--	--	--	--	--	N	Y
`pd_def_buf_control_area_assign`	Y	--	--	--	--	--	--	Y	Y
`pd_delete_reserved_word_file`#6	Y	--	--	--	--	--	--	Y	Y
`pd_deter_restart_on_stop_fail`	Y	--	--	--	--	--	--	Y	Y
`pd_dfw_awt_process`	--	--	Y	Y	--	Y	Y	Y	Y
`pd_dfw_syncpoint_skip_limit`	--	--	Y	Y	--	Y	Y	Y	Y
`pd_dic_shmpool_size`	--	--	Y	--	--	Y	--	Y#2	Y#2
`pd_down_watch_proc`	Y	Y	--	--	--	--	--	Y	Y
`pd_dump_suppress_watch_time`	Y	Y	--	--	--	--	--	Y	Y
`pd_fes_lck_pool_partition`	--	--	Y	--	Y	--	--	N	Y
`pd_fes_lck_pool_size`	--	--	Y	--	Y	--	--	Y	Y
`pd_floatable_bes`	--	--	--	--	Y	--	--	Y	Y
`pd_ha`	Y	--	--	--	--	--	--	N	N
`pd_ha_acttype`	--	Y	--	--	--	--	--	N	N
`pd_ha_agent`	--	Y	--	--	--	--	--	N	N
`pd_ha_ipaddr_inherit`	Y	Y	--	--	--	--	--	N	N
`pd_ha_max_act_guest_servers`	--	Y	--	--	--	--	--	N	N

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_ha_max_server_process	--	Y	--	--	--	--	--	N	Y[#2]
pd_ha_mgr_rerun	Y	--	--	--	--	--	--	Y	Y
pd_ha_process_count	--	Y	--	--	--	--	--	Y	Y
pd_ha_resource_act_wait_time	Y	Y	--	--	--	--	--	Y	Y
pd_ha_server_process_standby	--	Y	--	--	--	--	--	N	N
pd_ha_switch_timeout	Y	Y	--	--	--	--	--	Y	Y
pd_ha_transaction[#6]	Y	--	--	--	--	--	--	Y	Y
pd_ha_trn_queuing_wait_time	Y	--	--	--	--	--	--	Y	Y
pd_ha_trn_restart_retry_time	Y	--	--	--	--	--	--	Y	Y
pd_ha_unit	--	Y	--	--	--	--	--	N	N
pd_hash_table_size[#6]	Y	--	--	--	--	--	--	N	N
pd_hashjoin_hashing_mode[#6]	Y	--	--	--	--	--	--	Y	Y
pd_hostname	--	Y	--	--	--	--	--	N	Y
pd_host_watch_interval	Y	--	--	--	--	--	--	Y	Y
pd_indexlock_mode	Y	--	--	--	--	--	--	N	Y
pd_inet_unix_bufmode	Y	Y	--	--	--	--	--	Y	Y
pd_inet_unix_utl_bufmode	Y	Y	--	--	--	--	--	Y	Y
pd_ipc_clt_conn_nblock	Y	--	--	--	--	--	--	Y	Y
pd_ipc_clt_conn_nblock_time	Y	--	--	--	--	--	--	Y	Y
pd_ipc_conn_count	Y	--	--	--	--	--	--	Y	Y
pd_ipc_conn_interval	Y	--	--	--	--	--	--	Y	Y
pd_ipc_conn_nblock	Y	--	--	--	--	--	--	Y	Y
pd_ipc_conn_nblock_time	Y	--	--	--	--	--	--	Y	Y
pd_ipc_inet_bufsize	Y	Y	--	--	--	--	--	Y	Y
pd_ipc_recv_count	Y	Y	--	--	--	--	--	Y	Y
pd_ipc_send_count	Y	Y	--	--	--	--	--	Y	Y
pd_ipc_send_retrycount	Y	Y	--	--	--	--	--	Y	Y
pd_ipc_send_retrysleeptime	Y	Y	--	--	--	--	--	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_java_archive_directory	Y	Y	--	--	--	--	--	Y	Y
pd_java_castoff	--	--	Y	Y	Y	Y	Y	Y	Y
pd_java_classpath	Y	Y	--	--	--	--	--	Y	Y
pd_java_libpath	Y	Y	--	--	--	--	--	Y	Y
pd_java_option	Y	--	--	--	--	--	--	Y	Y
pd_java_routine_stack_size	Y	--	--	--	--	--	--	Y	Y
pd_java_runtimepath	Y	Y	--	--	--	--	--	Y	Y
pd_java_stdout_file	Y	Y	Y	Y	Y	Y	Y	Y	Y
pd_jp1_event_level	Y	--	--	--	--	--	--	Y	Y
pd_jp1_event_msg_out	Y	--	--	--	--	--	--	Y	Y
pd_jp1_use	Y	--	--	--	--	--	--	Y	Y
pd_key_resource_type	Y	--	--	--	--	--	--	N	Y
pd_large_file_use	Y	--	--	--	--	--	--	N	Y
pd_lck_deadlock_check	Y	Y	--	--	--	--	--	Y	Y
pd_lck_deadlock_check_interval	Y	Y	--	--	--	--	--	Y	Y
pd_lck_deadlock_info	Y	Y	--	--	--	--	--	Y	Y
pd_lck_hash_entry	--	--	Y	Y	Y	Y	Y	Y	Y
pd_lck_pool_partition	--	--	Y	Y	--	Y	Y	N	Y
pd_lck_pool_size	--	--	Y	Y	--	Y	Y	Y#2	Y#2
pd_lck_queue_limit	Y	--	--	--	--	--	--	Y	Y
pd_lck_release_detect	Y	Y	--	--	--	--	--	Y	Y
pd_lck_release_detect_interval	Y	Y	--	--	--	--	--	Y	Y
pd_lck_until_disconnect_cnt	--	--	Y	Y	--	Y	Y	N	Y
pd_lck_wait_timeout	Y	Y	--	--	--	--	--	Y	Y
pd_leap_second	Y	--	--	--	--	--	--	Y	Y
pd_list_initialize_timing	Y	--	--	--	--	--	--	Y	Y
pd_listen_socket_bufset	Y	Y	--	--	--	--	--	Y	Y
pd_lock_uncommited_delete_data	Y	--	--	--	--	--	--	N	I
pd_log_auto_expand_size	--	--	Y	Y	Y	Y	Y	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
`pd_log_auto_unload_path`	--	--	--	Y	Y	Y	Y	N	N
`pd_log_auto_unload_restart`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_dual`[#5]	--	--	Y	Y	Y	Y	Y	N	N
`pd_log_max_data_size`[#5]	--	--	Y	Y	Y	Y	Y	Y[#2, #3]	Y[#2]
`pd_log_rec_leng`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_remain_space_check`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_rerun_reserved_file_open`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_rerun_swap`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_rollback_buff_count`	--	--	Y	Y	Y	Y	Y	Y[#3]	Y
`pd_log_rpl_no_standby_file_opr`	Y	--	--	--	--	--	--	Y	Y
`pd_log_sdinterval`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_singleoperation`	--	--	Y	Y	Y	Y	Y	N	N
`pd_log_swap_timeout`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_unload_check`	--	--	Y	Y	Y	Y	Y	Y	Y
`pd_log_write_buff_count`	--	--	Y	Y	Y	Y	Y	Y[#3]	Y
`pd_master_file_name`	Y	--	--	--	--	--	--	N	N
`pd_max_access_tables`	Y	--	--	--	--	--	--	Y[#3]	Y
`pd_max_access_tables_wrn_pnt`	Y	--	--	--	--	--	--	Y	Y
`pd_max_add_dbbuff_no`	--	--	Y	Y	--	Y	Y	N	Y
`pd_max_add_dbbuff_shm_no`	--	--	Y	Y	--	Y	Y	N	Y
`pd_max_ard_process`	--	--	Y	Y	--	Y	Y	Y	Y
`pd_max_bes_process`	--	--	Y	--	--	--	Y	N	Y[#2]
`pd_max_commit_write_reclaim_no`	Y	--	--	--	--	--	--	Y	Y
`pd_max_dbbuff_shm_no`	Y	--	--	--	--	--	--	Y[#2, #7]	Y[#2, #7]
`pd_max_dic_process`	--	--	Y	--	--	Y	--	N	Y[#2]
`pd_max_file_no`	Y	--	--	--	--	--	--	Y[#2, #3]	Y
`pd_max_file_no_wrn_pnt`	Y	--	--	--	--	--	--	Y	Y
`pd_max_list_users`	Y	--	--	--	--	--	--	Y[#3]	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_max_list_count	Y	--	--	--	--	--	--	N	Y
pd_max_list_users_wrn_pnt	Y	--	--	--	--	--	--	Y	Y
pd_max_list_count_wrn_pnt	Y	--	--	--	--	--	--	Y	Y
pd_max_open_holdable_cursors	--	--	Y	Y	--	Y	Y	N	N
pd_max_rdarea_no	Y	--	--	--	--	--	--	N	Y
pd_max_rdarea_no_wrn_pnt	Y	--	Y	--	--	Y	--	Y	Y
pd_max_recover_process	Y	Y	--	--	--	--	--	Y	Y
pd_max_resident_rdarea_no	Y	--	--	--	--	--	--	Y	Y
pd_max_resident_rdarea_shm_no	Y	--	--	--	--	--	--	Y	Y
pd_max_server_process	Y	Y	--	--	--	--	--	Y	Y
pd_max_tmp_table_rdarea_no	Y	--	--	--	--	--	--	Y	Y
pd_max_temporary_object_no	Y	--	Y	Y	--	--	Y	Y	Y
pd_max_users	Y	--	--	--	--	--	--	N	Y
pd_max_users_wrn_pnt	Y	--	--	--	--	--	--	Y	Y
pd_mlg_file_size	Y	--	--	--	--	--	--	Y	Y
pd_mlg_msg_log_unit	Y	--	--	--	--	--	--	N	N
pd_mlg_port	Y	--	--	--	--	--	--	Y	Y
pd_mode_conf	Y	--	--	--	--	--	--	Y	Y
pd_module_trace_max	Y	Y	Y	Y	Y	Y	Y	Y	Y
pd_module_trace_timer_level	Y	Y	Y	Y	Y	Y	Y	Y	Y
pd_name_fixed_port_lookup	Y	--	--	--	--	--	--	Y	Y
pd_name_port	Y	--	--	--	--	--	--	I	I
pd_non_floatable_bes	--	--	--	--	Y	--	--	Y	Y
pd_nowait_scan_option	Y	--	--	--	--	--	--	Y	Y
pd_ntfs_cache_disable	Y	--	--	--	--	--	--	Y	Y
pd_oltp_holdcr	Y	--	--	--	--	--	--	Y	Y
pd_optimize_level#6	Y	--	--	--	Y	--	--	Y	Y
pd_overflow_suppress	Y	--	--	--	--	--	--	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_pageaccess_mode	Y	--	--	--	--	--	--	Y	Y
pd_plugin_ixmk_dir	--	--	--	Y	--	--	Y	N	Y
pd_prf_file_count	Y	Y	--	--	--	--	--	Y	Y
pd_prf_file_size	Y	Y	--	--	--	--	--	Y	Y
pd_prf_level	Y	Y	--	--	--	--	--	Y	Y
pd_prf_trace	Y	--	--	--	--	--	--	Y	Y
pd_process_count	--	--	Y	Y	Y	Y	Y	Y#2	Y#2
pd_process_desktopheap_size	Y	--	--	--	--	--	--	Y	Y
pd_process_terminator	Y	--	--	--	--	--	--	Y	Y
pd_process_terminator_max	Y	--	--	--	--	--	--	Y	Y
pd_pth_trace_max	Y	Y	Y	Y	Y	Y	Y	Y	Y
pd_queue_watch_time	Y	--	--	--	--	--	--	Y	Y
pd_queue_watch_timeover_action	Y	--	--	--	--	--	--	Y	Y
pd_rdarea_expand_format	Y	--	--	--	--	--	--	Y	Y
pd_rdarea_extension_timing	Y	--	--	--	--	--	--	Y	Y
pd_rdarea_list_no_wrn_pnt	Y	--	--	--	--	--	--	Y	Y
pd_rdarea_open_attribute	Y	--	--	--	--	--	--	Y	Y
pd_rdarea_open_attribute_use	Y	--	--	--	--	--	--	Y	Y
pd_rdarea_warning_point	Y	--	--	--	--	--	--	Y	Y
pd_rdarea_warning_point_msgout	Y	--	--	--	--	--	--	Y	Y
pd_redo_allpage_put	Y	--	--	--	--	--	--	Y	Y
pd_reduced_check_time	Y	--	--	--	--	--	--	Y	Y
pd_registered_port	Y	Y	--	--	--	--	--	N	I
pd_registered_port_check	Y	Y	--	--	--	--	--	N	I
pd_registered_port_level	Y	Y	--	--	--	--	--	N	Y
pd_registry_cache_size	--	--	Y	Y	Y	--	--	Y	Y
pd_rorg_predict	Y	--	--	--	--	--	--	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_routine_def_cache_size	--	--	Y	Y	Y	--	--	Y	Y
pd_rpc_bind_loopback_address	Y	--	--	--	--	--	--	Y	Y
pd_rpc_trace	Y	Y	Y	Y	Y	Y	Y	Y	Y
pd_rpc_trace_name	Y	Y	Y	Y	Y	Y	Y	Y	Y
pd_rpc_trace_size	Y	Y	Y	Y	Y	Y	Y	Y	Y
pd_rpl_func_control	Y	--	--	--	--	--	--	Y	Y
pd_rpl_hdepath	--	Y	--	--	--	--	--	N	N
pd_rpl_init_start	Y	--	--	--	--	--	--	N	N
pd_rpl_reflect_mode	Y	--	--	--	--	--	--	N	Y
pd_scd_port	Y	--	--	--	--	--	--	Y	Y
pd_sds_shmpool_size	--	--	Y	Y	--	--	--	Y#2	Y#2
pd_security_host_group	Y	--	--	--	--	--	--	Y	Y
pd_server_cleanup_interval	--	--	Y	Y	Y	Y	Y	Y	Y
pd_server_entry_queue	Y	Y	--	--	--	--	--	Y	Y
pd_service_port	Y	Y	--	--	--	--	--	I	I
pd_shared_memory_report	Y	--	--	--	--	--	--	Y	Y
pd_shared_rdarea_use	Y	--	--	--	--	--	--	N	Y
pd_shmpool_attribute	Y	Y	--	--	--	--	--	Y	Y
pd_shmpool_control	Y	--	--	--	--	--	--	Y	Y
pd_space_level#6	Y	--	--	--	--	--	--	Y	Y
pd_spd_assurance_count	--	--	Y	Y	Y	Y	Y	N	N
pd_spd_assurance_msg	--	--	Y	Y	Y	Y	Y	Y	Y
pd_spd_dual	--	--	Y	Y	Y	Y	Y	N	N
pd_spd_max_data_size	--	--	Y	Y	Y	Y	Y	N	N
pd_spd_reduced_mode	--	--	Y	Y	Y	Y	Y	N	Y
pd_spd_reserved_file_auto_open	--	--	Y	Y	Y	Y	Y	N	Y
pd_spd_syncpoint_skip_limit	--	--	Y	Y	Y	Y	Y	Y	Y
pd_spool_cleanup	Y	Y	--	--	--	--	--	Y	Y
pd_spool_cleanup_interval	Y	Y	--	--	--	--	--	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_spool_cleanup_interval_level	Y	Y	--	--	--	--	--	Y	Y
pd_spool_cleanup_level	Y	Y	--	--	--	--	--	Y	Y
pd_sql_command_exec_users	Y	--	--	--	--	--	--	Y	Y
pd_sql_dec_op_maxprec	Y	--	--	--	--	--	--	Y	Y
pd_sql_mode	Y	--	--	--	--	--	--	Y	Y
pd_sql_object_cache_size	Y	--	Y	Y	Y	Y	Y	N	Y
pd_sql_send_buff_size	Y	--	--	--	--	--	--	Y	Y
pd_sql_simple_comment_use	Y	--	--	--	--	--	--	Y	Y
pd_sqlobject_stat_timing	Y	--	--	--	--	--	--	Y	Y
pd_standard_sqlstate	Y	--	--	--	--	--	--	Y	Y
pd_start_level	Y	--	--	--	--	--	--	Y	Y
pd_start_skip_unit	Y	--	--	--	--	--	--	Y	Y
pd_start_time_out	Y	--	--	--	--	--	--	Y	Y
pd_statistics	Y	--	--	--	--	--	--	Y	Y
pd_stj_file_size	Y	Y	--	--	--	--	--	Y#3	Y
pd_stj_buff_size	Y	Y	--	--	--	--	--	Y#3	Y
pd_sts_file_name_1 to 7	--	--	--	Y	Y	Y	Y	N	N
pd_sts_initial_error	--	--	Y	Y	Y	Y	Y	Y	Y
pd_sts_last_active_file	--	--	--	Y	Y	Y	Y	Y	Y
pd_sts_last_active_side	--	--	--	Y	Y	Y	Y	Y	Y
pd_sts_singleoperation	--	--	Y	Y	Y	Y	Y	Y	Y
pd_substr_length	Y	--	--	--	--	--	--	Y	Y
pd_sysdef_default_option	Y	--	--	--	--	--	--	N	Y
pd_syssts_file_name_1 to 7	--	Y	--	--	--	--	--	N	N
pd_syssts_initial_error	--	Y	--	--	--	--	--	Y	Y
pd_syssts_last_active_file	--	Y	--	--	--	--	--	Y	Y
pd_syssts_last_active_side	--	Y	--	--	--	--	--	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
pd_syssts_singleoperation	--	Y	--	--	--	--	--	Y	Y
pd_system_complete_wait_time	Y	--	--	--	--	--	--	Y	Y
pd_system_dbsync_point	Y	--	--	--	--	--	--	N	Y
pd_system_id	Y	--	--	--	--	--	--	N	N
pd_table_def_cache_size	--	--	Y	Y	Y	--	--	Y#2, #3	Y#2
pd_tcp_inet_bufsize	Y	Y	--	--	--	--	--	Y	Y
pd_thdlock_pipe_retry_interval	Y	Y	--	--	--	--	--	Y	Y
pd_thdlock_retry_time	Y	Y	--	--	--	--	--	Y	Y
pd_thdlock_sleep_func	Y	--	--	--	--	--	--	Y	Y
pd_thdlock_wakeup_lock	Y	Y	--	--	--	--	--	Y	Y
pd_thdspnlk_spn_count	Y	Y	--	--	--	--	--	Y	Y
pd_thread_max_stack_size	Y	--	--	--	--	--	--	Y	Y
pd_thread_stack_expand_size	Y	Y	--	--	--	--	--	Y	Y
pd_tmp_directory	--	Y	--	--	--	--	--	Y	Y
pd_tmp_table_initialize_timing	Y	--	--	--	--	--	--	Y	Y
pd_trn_commit_optimize	Y	--	--	--	--	--	--	Y	Y
pd_trn_port	Y	--	--	--	--	--	--	Y	Y
pd_trn_rcvmsg_store_buflen	Y	Y	--	--	--	--	--	Y	Y
pd_trn_rerun_branch_auto_decide	Y	--	--	--	--	--	--	Y	Y
pd_trn_rollback_msg_interval	Y	--	--	--	--	--	--	Y	Y
pd_trn_rollback_watch_time	Y	--	--	--	--	--	--	Y	Y
pd_trn_send_decision_interval	Y	--	--	--	--	--	--	Y	Y
pd_trn_send_decision_intval_sec	Y	--	--	--	--	--	--	Y	Y
pd_trn_send_decision_retry_time	Y	--	--	--	--	--	--	Y	Y
pd_trn_watch_time	Y	--	--	--	--	--	--	Y	Y
pd_type_def_cache_size	--	--	Y	Y	Y	--	--	Y	Y

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
`pd_uap_exerror_log_dir`	Y	Y	--	--	--	--	--	Y	Y
`pd_uap_exerror_log_param_size`[#6]	Y	Y	--	--	--	--	--	Y	Y
`pd_uap_exerror_log_size`	Y	Y	--	--	--	--	--	Y	Y
`pd_uap_exerror_log_use`[#6]	Y	--	--	Y	Y	--	--	Y	Y
`pd_unit_id`	--	Y	--	--	--	--	--	N	N
`pd_utl_buff_size`	Y	--	--	--	--	--	--	Y[#2]	Y[#2]
`pd_utl_exec_mode`	Y	--	--	--	--	--	--	N	Y
`pd_utl_exec_time`	Y	--	--	--	--	--	--	Y	Y
`pd_utl_file_buff_size`	Y	--	--	--	--	--	--	Y	Y
`pd_view_def_cache_size`	--	--	Y	Y	Y	--	--	Y[#2, #3]	Y[#2]
`pd_watch_pc_client_time`[#6]	--	--	Y	Y	Y	--	--	Y	Y
`pd_watch_resource`	Y	--	--	--	--	--	--	Y	Y
`pd_watch_time`	Y	Y	--	--	--	--	--	Y	Y
`pd_work_buff_expand_limit`	--	--	Y	Y	--	Y	Y	Y	Y
`pd_work_buff_mode`	--	--	Y	Y	--	Y	Y	Y	Y
`pd_work_buff_size`	--	--	Y	Y	--	Y	Y	Y	Y
`pd_work_table_option`	Y	--	--	--	--	--	--	Y	Y
`pdaudload`	Y	--	--	--	--	--	--	Y	Y
`pdbuffer`	Y	--	--	--	--	--	--	N	Y
`pdcltgrp`	Y	--	--	--	--	--	--	Y	Y
`pdhagroup`	Y	--	--	--	--	--	--	N	N
`pdhibegin`	Y	--	--	--	--	--	--	Y	Y
`pdlogadfg -d spd`	--	--	--	Y	Y	Y	Y	Y[#1]	Y[#1]
`pdlogadfg -d sys`[#5]	--	--	--	Y	Y	Y	Y	Y[#1]	Y[#1]
`pdlogadpf -d spd`	--	--	--	Y	Y	Y	Y	Y[#1]	Y[#1]
`pdlogadpf -d sys`[#5]	--	--	--	Y	Y	Y	Y	Y[#1]	Y[#1]
`pdmlgput`	Y	--	--	--	--	--	--	Y	Y
`pdplgprm`	--	--	--	Y	Y	Y	Y	Y	Y
`pdplugin`	Y	--	--	--	--	--	--	N	N
`pdstart`	Y	--	--	--	--	--	--	N	Y[#8]

Operand	Definition							MOD FRCD?	MOD PLND?
	SYS	UNT	SVR	SDS	FES	DS	BES		
`pdstbegin`	Y	--	--	--	--	--	--	Y	Y
`pdunit`	Y	--	--	--	--	--	--	N	Y#8
`pdwork`	--	--	--	Y	--	Y	Y	N	N
`pdwork_wrn_pnt`	Y	--	--	--	--	--	--	Y	Y
`SHMMAX`	Y	Y	--	--	--	--	--	Y#7	Y#7
`TZ`	Y	--	--	--	--	--	--	Y	Y

Legend:

MOD FRCD?: Modifiable after forced or abnormal termination?

MOD PLND?: Modifiable after planned termination?

Y: Can be modified.

N: Cannot be modified.

I: Typically needs to be modified, but be careful to maintain system integrity.

--: Not applicable

SYS: System common definition

UNT: Unit control information definition

SVR: Server common definition

SDS: Single server definition

FES: Front-end server definition

DS: Dictionary server definition

BES: Back-end server definition

#1

This operand can be added if it does not exist; if it already exists, it cannot be deleted or modified.

#2

It might not be possible to restart HiRDB if the value specified in this operand is decreased. If this occurs, revert the specification to the value prior to the change, and then restart.

#3

It might not be possible to restart HiRDB if the value specified in this operand is increased. If this occurs, revert the specification to the value prior to the change, and then restart.

#4

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, HiRDB will start with the value in effect prior to modification, even if the definition has been modified at the time of the HiRDB restart.

#5

Exercise caution in modifying this operand while connected to HiRDB Datareplicator (source). Stop the corresponding HiRDB Datareplicator before adding, modifying, or deleting the specification of this operand, and while restarting HiRDB after such a change. Changing this definition and restarting HiDRB while HiRDB Datareplicator is running might result in failure of extraction by HiRDB Datareplicator.

#6

The values of the HiRDB system definition operands shown below can be modified in the client environment definitions of the individual clients. To modify a value for a client, specify the applicable client environment definition operand. For details about the client environment definitions, see the *HiRDB Version 9 UAP Development Guide*.

HiRDB system definition operand	Client environment definition operand
`pd_additional_optimize_level`	`PDADDITIONALOPTLVL`

HiRDB system definition operand	Client environment definition operand
pd_cwaittime_wrn_pnt	PDCWAITTIMEWRNPNT
pd_delete_reserved_word_file	PDDELRSVWDFILE
pd_ha_transaction	PDHATRNQUEUING
pd_hash_table_size	PDHASHTBLSIZE
pd_hashjoin_hashing_mode	PDHJHASHINGMODE
pd_lck_wait_timeout	PDLCKWAITTIME
pd_optimize_level	PDSQLOPTLVL
pd_space_level	PDSPACELVL
pd_uap_exerror_log_param_size	PDUAPEXERLOGPRMSZ
pd_uap_exerror_log_use	PDUAPEXERLOGUSE
pd_watch_pc_client_time	PDSWATCHTIME

#7

If `pd_dbbuff_modify=Y` is specified in the system common definition (`pdsys`), this specification cannot be modified.

#8

Whether the operand can be modified depends on which options are specified. For details, see the descriptions of the individual operands.

1.4.2 Operand priorities

If the same operand is defined in different definitions, HiRDB determines the value to be used on the basis of the priority order of the definitions, as shown below:

1. Single server definition, front-end server definition, dictionary server definition, back-end server definition
2. Server common definition
3. Unit control information definition
4. System common definition

A standard value can be specified in a definition that has a low priority, and then that value can be changed by specifying a different value in a definition with a higher priority.

If an operand is omitted from all definitions, the default value for the lowest priority definition is assumed.

1.5 Operands whose default value depends on the version and operands that are no longer needed

(1) Operands whose default value depends on the version

The table below lists and describes the operands whose default value depends on the version.

If you do not want to change operand default values during upgrading, specify `v6compatible` or `v7compatible` in the `pd_sysdef_default_option` operand.

Table 1–8: Operands whose default value depends on the version

Operand name	Default value in 08-00 and later versions (recommendable#1)	Default value in 07-00 (v7compatible#2)	Default value in versions earlier than 07-00 (v6compatible#3)
`pd_additional_optimize_level`	`"COST_BASE_2"`	`"COST_BASE_2"`	`"NONE"`
`pd_assurance_index_no`	`500`	• For HiRDB/Single Server: Number of indexes for data dictionary table (240) + 50 • For HiRDB/Parallel Server: `50`	• For HiRDB/Single Server: Number of indexes for data dictionary table (240) + 50 • For HiRDB/Parallel Server: `50`
`pd_assurance_table_no`	`500`	`100`	`0`
`pd_aud_async_buff_count`	MAX(1, number of HiRDB servers in unit × 10)	`3`	`3`
`pd_aud_async_buff_size`	`401408`	`4096`	`4096`
`pd_bes_shmpool_size`	Automatic computation by HiRDB	Automatic computation by HiRDB	`1024`
`pd_cancel_down_msgchange`	`Y`	`N`	`N`
`pd_check_pending`#4	USE	• 07-03 and later versions: `USE` • Versions earlier than 07-03: `NOUSE` (`USE`#5)	`NOUSE` (`USE`#5)
`pd_change_clt_ipaddr`	`1`	`0`	`0`
`pd_dbbuff_lru_option`	`MIX`	`SEPARATE`	`SEPARATE`
`pd_dic_shmpool_size`	Automatic computation by HiRDB	Automatic computation by HiRDB	`1024`
`pd_large_file_use`	• 08-05 and later versions: `Y` • Versions earlier than 08-05: `N`	`N`	`N`
`pd_lck_hash_entry`	`0` (Automatic computation by HiRDB)	`0` (Automatic computation by HiRDB)	`11261`
`pd_lck_pool_size`	`16000` (`32000` in the 64-bit mode)	`16000` (`32000` in the 64-bit mode)	`1024`
`pd_lck_wait_timeout`	MAX(`180`, *value of pd_watch_time*)	MAX(`180`, *value of pd_watch_time*)	`180`

Operand name	Default value in 08-00 and later versions (recommendable[#1])	Default value in 07-00 (v7compatible[#2])	Default value in versions earlier than 07-00 (v6compatible[#3])
`pd_log_max_data_size`	`400000`	`32000`	`32000`
`pd_log_sdinterval`	• *system-log-output-volume*: `5000` • Interval: `60`	• *system-log-output-volume*: `1000` • Interval: `60`	• *system-log-output-volume*: `1000` • Interval: `60`
`pd_log_write_buff_count`	`10`	`3`	`3`
`pd_max_access_tables`	`64`	`64`	`16`
`pd_max_commit_write_reclaim_no`	`10`	`0`	`0`
`pd_max_server_process`	Automatic computation by HiRDB	Automatic computation by HiRDB	`100`
`pd_ntfs_cache_disable`	Y	N	N
`pd_optimize_level`	Refer to the `pd_optimize_level` operand.	Refer to the `pd_optimize_level` operand.	`"SELECT_APSL"`
`pd_pageaccess_mode`	`SNAPSHOT`	`SNAPSHOT`	`NORMAL`
`pd_process_terminator`	`Fixed`	`fixed`	`nonresident`
`pd_process_terminator_max`	Max(3, ↑ *value of pd_max_users operand* ÷ 100 ↑)	↑ *value of pd_max_users operand* ÷ 100 ↑	↑ *value of pd_max_users operand* ÷ 100 ↑
`pd_sds_shmpool_size`	Automatic computation by HiRDB	Automatic computation by HiRDB	`1024`
`pd_spool_cleanup_interval_level`	• *number-of-days*: `7` • *deletion-type*: `all`	• *number-of-days*: `7` • *deletion-type*: `dump`	• *number-of-days*: `7` • *deletion-type*: `dump`
`pd_spool_cleanup_level`	• *number-of-days*: `7` • *deletion-type*: `all`	• *number-of-days*: `7` • *deletion-type*: `dump`	• *number-of-days*: `7` • *deletion-type*: `dump`
`pd_thdlock_pipe_retry_interval`	`1000000`	`1000000`	`4000000` (`1000000`[#6])
`pd_thdlock_retry_time`	`1000`	`10000`	`10000`
`pd_trn_commit_optimize`	`ONEPHASE`	`ONEPHASE`	`NOUSE`
`pd_trn_send_decision_interval`	Value of `pd_trn_send_decision_intval_sec` operand	5	5
`pd_work_buff_mode`	`Pool`	`pool`	`each`
`pd_work_table_option`	`1`	`1`	`0`
`SHMMAX`	`200` (only in the 32-bit mode)	`200` (only in the 32-bit mode)	`6`

#1

These are the default values when `recommendable` is specified in the `pd_sysdef_default_option` operand or when the `pd_sysdef_default_option` operand has been omitted.

#2

These are the default values used when `v7compatible` is specified in the `pd_sysdef_default_option` operand.

#3

These are the default values used when `v6compatible` is specified in the `pd_sysdef_default_option` operand.

#4

The default for the `pd_check_pending` operand was changed in version 07-03 from `NOUSE` to `USE`.

#5

The `pd_check_pending` operand is not affected when the default value is modified with the `pd_sysdef_default_option` operand. For this reason, the default for `pd_check_pending` is `USE` even when `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand.

If compatibility with versions earlier than 07-03 must be maintained, specify `NOUSE` in `pd_check_pending`.

#6

The `pd_thdlock_pipe_retry_interval` operand is not affected when the default value is modified with the `pd_sysdef_default_option` operand. For this reason, the default for `pd_thdlock_pipe_retry_interval` is `1000000` even when `v6compatible` is specified in the `pd_sysdef_default_option` operand.

If compatibility with versions earlier than 07-00 must be maintained, specify `4000000` in `pd_thdlock_pipe_retry_interval`.

(2) Operands that are no longer needed

Upgrading eliminates the need to specify the operands listed below. If you have upgraded your system and have left these operands in place, no error occurs.

Operand that is no longer needed starting in version 08-00

- `pd_dynamic_sql_object_cache`

Operands that are no longer needed starting in version 07-00

- `pd_multi_fes`
- `pd_redo_skip_inf`

2 System Common Definition

This chapter explains the operands of the system common definition.

2.1 Operand formats

The system common definition defines the structure of the overall HiRDB system, as well as information that is common to all HiRDB system units. This section explains the formats used to specify the operands of a system common definition. Note that the numbers in the following table correspond to the numbers assigned to the explanations of the individual operands in *2.2 Operand explanations*.

For users who are creating HiRDB system definitions for the first time

The first step is to determine the values to be specified for the operands shown in boldface type. In principle, HiRDB can be started once the boldface operands have been specified.

No.	Format	Operand category
1	**set pd_system_id = *HiRDB-identifier***	*System structure*
2	**[set pd_name_port = *HiRDB-port-number*]**	
3	**set pd_master_file_name = *"HiRDB-file-name-at-beginning-of-master-directory-RDAREA"***	
4	**[set pd_max_users = *maximum-number-of-concurrent-connections*]**	*Maximum concurrent executions*
5	**[set pd_max_server_process = *maximum-number-of-concurrently-activated-server-processes*]**	
6	**[set pd_max_access_tables = *concurrently-accessible-base-tables-count*]**	
7	[set pd_utl_exec_mode = 0 | 1]	
8	[set pd_max_commit_write_reclaim_no = *maximum-number-of-concurrent-executions-of-pdreclaim-commands-with-p-option-specified*]	
9	[set pd_mode_conf = AUTO | MANUAL1 | MANUAL2]	*HiRDB startup*
10	[set pd_system_complete_wait_time = *pdstart-command-completion-wait-time*]	
11	[set pd_start_time_out = *HiRDB-start-preparation-maximum-wait-time*]	
12	[set pd_start_level = 0 | 1]	*Reduced activation*
13	[set pd_reduced_check_time = *wait-time-for-reduced-activation-startup-notice*]	
14	[set pd_start_skip_unit = *name-of-unit-not-to-be-started* [,*name-of-unit-not-to-be-started*] ...]	
15	[set pd_dbsync_point = sync | commit]	*HiRDB processing*
16	[set pd_system_dbsync_point = sync | commit]	
17	[set pd_dbsync_altwrite_skip = Y | N]	
18	[set pd_process_terminator = resident | fixed | nonresident]	
19	[set pd_process_terminator_max = *maximum-number-of-resident-post-processing-processes*]	
20	[set pd_process_desktopheap_size = *size-of-desktop-heap-used-per-process*]	
21	[set pd_server_entry_queue = spnfifo | fifo | loop]	
22	[set pd_thdlock_sleep_func = 0 | 1]	
23	[set pd_thdlock_wakeup_lock = Y | N]	
24	[set pd_thdlock_pipe_retry_interval = *thread-lock-release-check-interval*]	
25	[set pd_thdlock_retry_time = *thread-lock-sleep-time*]	
26	[set pd_thdspnlk_spn_count = *thread-spin-lock-spin-count*]	

No.	Format	Operand category
27	[set pd_pageaccess_mode = SNAPSHOT | NORMAL]	
28	[set pd_cmdhold_precheck = Y | N]	
29	[set pd_db_io_error_action = dbhold | unitdown]	
30	[set pd_connect_errmsg_hide = Y | N]	
31	[set pd_rpc_bind_loopback_address = Y |N | S]	
32	[set pd_cancel_down_msgchange = Y | N]	
33	[set pd_max_recover_process = *concurrently-executable-full-recovery-processes-count*]	*Full recovery processing*
34	[set pd_redo_allpage_put= Y | N]	
35	[set pd_trn_rerun_branch_auto_decide = Y | N]	*Transaction decision processing*
36	[set pd_trn_send_decision_intval_sec = *transmission-retry-interval-in-seconds-for-automatic-transaction-decision*]	
37	[set pd_trn_send_decision_interval = *transmission-retry-interval-in-minutes-for-automatic-transaction-decision*]	
38	[set pd_trn_send_decision_retry_time = *maximum-wait-time-for-transaction-auto-decision*]	
39	[set pd_trn_watch_time = *maximum-communication-wait-time-during-transaction-synchronization-point-processing*]	
40	[set pd_trn_rcvmsg_store_buflen = *transaction-recovery-message-queue-size*]	
41	[set pd_trn_commit_optimize = ONEPHASE | NOUSE]	
42	[set pd_trn_rollback_msg_interval = *interval-at-which-a-message-indicating-that-rollback-is-underway-is-output*]	
43	[set pd_trn_rollback_watch_time = *maximum-wait-time-for-rollback-completion-response* [, *rollback-instruction-resending-limit-time*]]	
44	[set pd_overflow_suppress = Y | N]	*SQL specifications*
45	[set pd_space_level = 0 | 1 | 3]	
46	[set pd_dec_sign_normalize = Y | N]	
47	[set pd_sql_dec_op_maxprec = *maximum-precision-for-a-DECIMAL-type-operation-result-not-exceeding-29-digits*]	
48	[set pd_sql_mode = 0 | 1]	
49	[set pd_sql_simple_comment_use = Y | N]	
50	[set pd_optimize_level = *SQL-optimization-option* [,*SQL-optimization-option*] ...]	*SQL optimization*
51	[set pd_additional_optimize_level = *SQL-extension-optimizing-option* [,*SQL-extension-optimizing-option*] ...]	
52	[set pd_hashjoin_hashing_mode = TYPE1 | TYPE2]	
53	[set pd_hash_table_size = *hash-table-size*]	
54	[set pd_work_table_option = *work-table-processing-option*]	
55	[set pd_apply_search_ats_num = *maximum-number-of-combinations-of-narrowing-values-applied-to-ATS-search-condition*]	
56	[set pd_max_list_users = *number-of-users-who-can-own-lists-concurrently*]	*Narrowed retrieval*

No.	Format	Operand category
57	[set pd_max_list_count = *number-of-lists-created-per-user*]	
58	[set pd_list_initialize_timing = INITIAL | DEFER | STANDBY]	
59	[set pd_utl_exec_time = *utility-execution-monitoring-time*]	*System monitoring*
60	[set pd_watch_time = *maximum-response-wait-time*]	
61	[set pd_queue_watch_time = *message-queue-monitoring-time*]	
62	[set pd_queue_watch_timeover_action = continue | stop]	
63	[set pd_down_watch_proc = *upper-limit-for-server-process-abnormal-terminations* [, *monitoring-interval*]]	
64	[set pd_host_watch_interval = *host-to-host-monitoring-interval*]	
65	[set pd_watch_resource = MANUAL | AUTO]	
66	[set pd_max_users_wrn_pnt = *trigger-for-outputting-warning-message-related-to-number-of-connections-to-HiRDB-server* [, *trigger-for-resetting-warning-message-output-status*]]	
67	[set pd_max_access_tables_wrn_pnt = *trigger-for-issuing-concurrently-accessible-base-tables-count-warning*]	
68	[set pd_max_rdarea_no_wrn_pnt = *trigger-for-issuing-RDAREAs-count-warning*]	
69	[set pd_max_file_no_wrn_pnt = *trigger-for-issuing-HiRDB-files-count-warning*]	
70	[set pdwork_wrn_pnt = *trigger-for-issuing-work-table-files-warning*]	
71	[set pd_max_list_users_wrn_pnt = *trigger-for-issuing-warning-about-number-of-users-who-have-created-lists*]	
72	[set pd_max_list_count_wrn_pnt = *trigger-for-issuing-warning-about-number-of-lists-created-by-a-user*]	
73	[set pd_rdarea_list_no_wrn_pnt = *trigger-for-issuing-warning-about-number-of-lists-created-at-server* [,*trigger-for-resetting-warning-output-status*]]	
74	[set pd_cwaittime_wrn_pnt = *output-condition-for-SQL-runtime-warning-information (% specification)* | *output-condition-for-SQL-runtime-warning-information (time specification)*]	*SQL runtime warning output facility*
75	[set pd_cwaittime_report_dir = *SQL-runtime-warning-information-file-output-destination-directory*]	
76	[set pd_cwaittime_report_size = *SQL-runtime-warning-information-file-maximum-size*]	
77	[set pd_uap_exerror_log_use = YES | NO]	*Facility for output of extended SQL error information*
78	[set pd_uap_exerror_log_dir = *SQL-error-report-file-storage-directory*]	
79	[set pd_uap_exerror_log_size = *SQL-error-report-file-maximum-size*]	
80	[set pd_uap_exerror_log_param_size = *maximum-data-size-of-parameter-information-to-be-output-to-client-error-log-file-and-SQL-error-report-file*]	
81	[set pd_delete_reserved_word_file = *SQL-reserved-word-deletion-file-name-1* [, *SQL-reserved-word-deletion-file-name-2*] ...]	*SQL reserved word deletion facility*
82	[set pd_sql_command_exec_users = *authorization-identifier* [, *authorization-identifier*] ...]	*Command execution from SQL*
83	[set pd_standard_sqlstate = Y | N]	*Detailed SQLSTATE values*
84	[set pd_lck_deadlock_info = Y | N]	*Lock*

No.	Format	Operand category
85	[set pd_lck_wait_timeout = *lock-release-wait-time*]	
86	[set pd_lck_release_detect = interval | pipe]	
87	[set pd_lck_release_detect_interval = *lock-release-detection-interval*]	
88	[set pd_nowait_scan_option = LOCK | NOLOCK]	
89	[set pd_lck_queue_limit = *trigger-for-issuing-warning-about-number-of-users-waiting-for-lock-release*]	
90	[set pd_deadlock_priority_use = Y | N]	
91	[set pd_command_deadlock_priority = 32 | 64 | 96 | 120]	
92	[set pd_key_resource_type = TYPE1 | TYPE2]	
93	[set pd_indexlock_mode = {KEY | NONE}]	
94	[set pd_lock_uncommited_delete_data = WAIT | NOWAIT]	
95	[set pd_dbreuse_remaining_entries = ALL | NONE]	
96	[set pd_lck_deadlock_check = Y | N]	
97	[set pd_lck_deadlock_check_interval = *deadlock-check-interval*]	
98	**[set pd_sql_object_cache_size = *SQL-object-buffer-size*]**	*Buffers*
99	[set pd_def_buf_control_area_assign = INITIAL | TRAN]	
100	[set pd_thread_max_stack_size = *maximum-stack-size-for-use-by-one-thread*]	
101	[set pd_thread_stack_expand_size = *stack-extension-size-per-thread*]	
102	[set pd_shmpool_attribute = free | fixed]	*Shared memory*
103	[set pd_shmpool_control = unit | server]	
104	[set pd_dbbuff_attribute = free | fixed]	
105	[set pd_shared_memory_report = Y | N]	
106	[set pd_max_dbbuff_shm_no = *maximum-number-of-shared-memory-segments-for-global-buffer*]	
107	[set pd_mlg_msg_log_unit = manager | local]	*Message log files*
108	[set pd_mlg_file_size = *maximum-message-log-file-size*]	
109	[set pd_statistics = Y | N]	*Statistical information*
110	[set pd_stj_file_size = *maximum-statistics-log-file-size*]	
111	[set pd_stj_buff_size = *statistics-log-buffer-size*]	
112	[set pd_sqlobject_stat_timing = deallocate | tran]	
113	[set pd_rpc_trace = Y | N]	*RPC trace information*
114	[set pd_rpc_trace_name = *"name-for-RPC-trace-collection-files"*]	
115	[set pd_rpc_trace_size = *RPC-trace-collection-file-size*]	
116	[set pd_prf_trace = Y | N]	*Performance trace information*
117	[set pd_prf_level = 00000007 | 0000001f | 0000007f | 000001ff | 00000000]	

No.	Format	Operand category
118	[set pd_prf_file_count = *number-of-performance-trace-information-file-generations*]	
119	[set pd_prf_file_size = *size-of-each-performance-trace-information-file*]	
120	[set pd_cancel_dump = put | noput]	*Troubleshooting information*
121	[set pd_client_waittime_over_abort = Y | N]	
122	[set pd_clt_waittime_over_dump_level = all | shm_fesonly]	
123	[set pd_dump_suppress_watch_time = *troubleshooting-information-output-suppression-time*]	
124	[set pd_debug_info_netstat = Y | N]	
125	[set pd_spool_cleanup_interval = *troubleshooting-information-deletion-interval*]	
126	[set pd_spool_cleanup_interval_level = *number-of-days* [, *deletion-type*]]	
127	[set pd_spool_cleanup = normal | force | no]	
128	[set pd_spool_cleanup_level = *number-of-days* [,*deletion-type*]]	
129	[set pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*]	
130	[set pd_module_trace_timer_level = 0 | 10 | 20]	
131	[set pd_pth_trace_max = *maximum-number-of-stored-communication-traces*]	
132	**[set pd_max_rdarea_no =** *maximum-number-of-RDAREAs***]**	*RDAREAs*
133	**[set pd_max_file_no =** *maximum-number-of-HiRDB-files-comprising-an-RDAREA***]**	
134	[set pd_rdarea_warning_point = *segment-usage-ratio-1* [,*segment-usage-ratio-2* [,*segment-usage-ratio-3*]]]	
135	[set pd_rdarea_warning_point_msgout = Y | N]	
136	[set pd_rdarea_extension_timing = use | nouse]	
137	[set pd_rdarea_expand_format = Y | N]	
138	[set pd_rdarea_open_attribute_use = Y | N]	
139	[set pd_rdarea_open_attribute = INITIAL | DEFER | SCHEDULE]	
140	[set pd_shared_rdarea_use = Y | N]	
141	[set pd_db_access_error_action = dbhold | unitdown]	
142	[set pd_db_hold_action = dbhold | unitdown]	
143	[set pd_dbbuff_lru_option = SEPARATE | MIX]	*Global buffers*
144	[set pd_dbbuff_binary_data_lru = Y | N]	
145	[set pd_dbbuff_modify = Y | N]	
146	[set pd_dbbuff_lock_release_detect = pipe | interval | switch]	
147	[set pd_dbbuff_lock_spn_count = *number-of-spins-during-lock-acquisition-wait-processing*]	
148	[set pd_dbbuff_lock_interval = *interval-during-lock-acquisition-wait-processing*]	
149	[set pd_dbbuff_wait_interval = *global-buffer-occupation-state-check-interval*]	

No.	Format	Operand category
150	`[set pd_dbbuff_wait_spn_count =` *maximum-spin-loop-count-for-global-buffer-occupation-state-checking*`]`	
151	`[set pd_dbbuff_rate_updpage =` *deferred-write-trigger-request-rate*`]`	
152	`[set pd_dbbuff_trace_level =` *global-buffer-control-information-trace-acquisition-level*`]`	
153	`[set pd_max_resident_rdarea_no =` *maximum-number-of-in-memory-RDAREAs*`]`	*In-memory data processing*
154	`[set pd_max_resident_rdarea_shm_no =` *maximum-number-of-shared-memory-segments-used-by-in-memory-data-buffer*`]`	
155	`[set pd_assurance_table_no =` *table-reservation-count*`]`	*Table or index reservation count*
156	`[set pd_assurance_index_no =` *index-reservation-count*`]`	
157	`[set pd_constraint_name = LEADING | TRAILING]`	*Referential and check constraints*
158	`[set pd_check_pending = USE | NOUSE]`	
159	`[set pd_max_tmp_table_rdarea_no =` *maximum-number-of-temporary-table-RDAREAs*`]`	*Temporary tables*
160	`[set pd_max_temporary_object_no =` *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*`]`	
161	`[set pd_tmp_table_initialize_timing = STARTING | ACCESS]`	
162	`[set pd_large_file_use = Y | N]`	*HiRDB file system areas*
163	`[set pd_ntfs_cache_disable = Y | N]`	
164	`[set pd_rorg_predict = Y | N]`	*Facility for predicting reorganization time*
165	`[set pd_audit = Y | N]`	*Security*
166	`[set pd_aud_file_name =` *HiRDB-file-system-area-name-for-audit-trail-file*`]`	
167	`[set pd_aud_max_generation_size =` *audit-trail-file-maximum-size*`]`	
168	`[set pd_aud_max_generation_num =` *maximum-audit-trail-file-count*`]`	
169	`[set pd_aud_no_standby_file_opr = down | forcewrite]`	
170	`[set pd_aud_async_buff_size =` *size-of-buffer-used-for-asynchronous-output-of-audit-trail-file*`]`	
171	`[set pd_aud_async_buff_count =` *number-of-buffer-sectors-used-for-asynchronous-output-of-audit-trail-file*`]`	
172	`[set pd_aud_async_buff_retry_intvl =` *retry-interval-for-allocation-of-a-buffer-to-be-used-for-asynchronous-output-of-audit-trail-file*`]`	
173	`[set pd_aud_sql_source_size =` *size-of-SQL-statement-output-to-audit-trail*`]`	
174	`[set pd_aud_sql_data_size =` *size-of-SQL-data-output-to-audit-trail*`]`	
175	`[set pd_aud_file_wrn_pnt =` *warning-message-output-trigger*`[,`*trigger-for-resetting-warning-message-output-status*`]]`	
176	`[set pd_aud_auto_loading = Y | N]`	

177	`[pdaudload[-i` *index-creation-method*`]`
	`[-l` *log-acquisition-mode*`]`
	`[-n [`*batch-output-local-buffer-sector-count*`],,`
	`[`*random-access-local-buffer-sector-count*`]]`	

No.	Format	Operand category
	[-y] [-X *response-monitoring-time-for-server-to-server-communication*] [-S *sort-buffer-size*]]	
178	[set pd_security_host_group = *host-name* [, *host-name*] ...]	
179	[set pd_ha = use | nouse]	*System switchover facility*
180	[set pd_ha_ipaddr_inherit = Y | N]	
181	[set pd_ha_switch_timeout = Y | N]	
182	[set pd_ha_mgr_rerun = wait | notwait]	
183	[set pd_ha_transaction = error | queuing]	
184	[set pd_ha_trn_queuing_wait_time = *transaction-queuing-wait-time*]	
185	[set pd_ha_trn_restart_retry_time = *retry-time-upper-limit-after-transaction-start-request-errors*]	
186	[set pd_ha_resource_act_wait_time = *maximum-wait-time-for-resource-activation*]	
187	[set pd_deter_restart_on_stop_fail = Y | N]	
188	[set pd_rpl_init_start = Y | N]	*HiRDB Datareplicator*
189	[set pd_rpl_reflect_mode = server | uap]	
190	[set pd_log_rpl_no_standby_file_opr = stop | continue]	
191	[set pd_rpl_func_control = BACKWARD_CUTOFF_UPDATE | NONE]	
192	[set pd_jp1_use = Y | N]	*Linkage to JP1*
193	[set pd_jp1_event_level = 1 | 2]	
194	[set pd_jp1_event_msg_out = Y | N]	
195	[set pd_oltp_holdcr = use | nouse]	*OLTP*
196	[set pd_auto_vrup = Y | N]	*Version upgrade*
197	[set pd_sysdef_default_option = recommendable | v6compatible | v7compatible]	
198	[set pd_service_port = *scheduler-process-port-number*]	*Communication processing*
199	[set pd_name_fixed_port_lookup = Y | N]	
200	[set pd_scd_port = *scheduler-process-port-number*]	
201	[set pd_trn_port = *transaction-server-process-port-number*]	
202	[set pd_mlg_port = *message-log-server-process-port-number*]	
203	[set pd_alv_port =*unit-monitoring-process-port-number*]	
204	[set pd_change_clt_ipaddr = 0 | 1]	
205	[set pd_registered_port = *"port-number-reservation-range"* [,*"port-number-reservation-range"*] ...]	
206	[set pd_registered_port_check = Y | N | C | W]	
207	[set pd_registered_port_level = 0 | 1]	

No.	Format	Operand category
208	`[set pd_ipc_send_retrycount =` *process-to-process-send-retries-count*`]`	
209	`[set pd_ipc_send_retrysleeptime =` *process-to-process-send-retry-sleep-time*`]`	
210	`[set pd_ipc_send_count =` *server-to-server-send-retries-count*`]`	
211	`[set pd_ipc_recv_count =` *server-to-server-receive-retries-count*`]`	
212	`[set pd_ipc_conn_nblock = Y | N]`	
213	`[set pd_ipc_conn_nblock_time =` *connection-establishment-monitoring-time-in-non-block-mode*`]`	
214	`[set pd_ipc_clt_conn_nblock = Y | N]`	
215	`[set pd_ipc_clt_conn_nblock_time =` *connection-establishment-monitoring-interval-in-non-block-mode-(communication-with-HiRDB-clients)*`]`	
216	`[set pd_ipc_conn_interval =` *connection-establishment-retry-interval*`]`	
217	`[set pd_ipc_conn_count =` *connection-establishment-retry-count*`]`	
218	`[set pd_inet_unix_bufmode = os | conf]`	
219	`[set pd_ipc_inet_bufsize =` *send-receive-buffer-size-for-server-to-server-communication*`]`	
220	`[set pd_tcp_inet_bufsize =` *send-receive-buffer-size-for-communication-with-HiRDB-client*`]`	
221	`[set pd_listen_socket_bufset = 0 | 1]`	
222	`[set pd_utl_buff_size =` *utility-communication-buffer-size*`]`	
223	`[set pd_inet_unix_utl_bufmode = auto | conf]`	
224	`[set pd_utl_file_buff_size =` *utility-file-buffer-size*`]`	
225	`[set pd_sql_send_buff_size =` *buffer-size-for-inter-server-communication-for-SQL-execution*`]`	
226	`[set pd_java_option =` *"Java-option"* `[,`*"Java-option"*`]... ]`	*Java*
227	`[set pd_java_routine_stack_size =` *stack-size-for-use-by-external-Java-routine*`]`	
228	`[set pd_java_archive_directory =` *"JAR-file-storage-directory"*`]`	
229	`[set pd_java_classpath =` *"Java-class-path"*`]`	
230	`[set pd_java_runtimepath =` *"Java-Runtime-Environment-root-directory"*`]`	
231	`[set pd_java_libpath =` *"Java-virtual-machine-library-directory"*`]`	
232	`[set pd_java_stdout_file =` *"Java-virtual-machine-standard-output-and-standard-error-output-destination-file"*`]`	
233	`[set pd_c_library_directory = "`*C-library-file-storage-directory*`"]`	*External C stored routine*
234	`[set pd_substr_length = 3 | 4 | 5 | 6]`	*Character encoding*
235	`[set pd_leap_second = Y | N]`	*Date and time*

236	**`{{ pdunit -x`** ***`host-name`***
	`-u` ***`unit-identifier`***
	`[-d "`*`HiRDB-directory-name`*`"]`
	`[-c` ***`host-name`*****`]`**
	
	[-p *HiRDB-port-number*]
		Unit structure

No.	Format	Operand category
	[-s *scheduler-process-port-number*] [-t *transaction-server-process-port-number*] [-m *message-log-server-process-port-number*] [-a *unit-monitoring-process-port-number*]}}	
237	{{ pdstart -t *server-type* [-s *server-name*] -x *host-name* | -u *unit-identifier* [-m *host-name*[,*host-name*]... [-n *host-name*[,*host-name*]...]] [-c *server-name* | -g *HA-group-identifier*] [-k stls]}}	*Server structure*
238	[{{ pdbuffer -a *global buffer-name* {-r *RDAREA-name*[,*RDAREA-name*]... | -b *RDAREA-name* [,*RDAREA-name*]... | -o | -i *authorization-identifier.index-identifier*} -n *buffer-sectors-count*[-l *buffer-size*] [-m *maximum-concurrently-executable-prefetches-count*] [-p *maximum-batch-input-pages-count*] [-w *updated-pages-output-rate-during-deferred-write-trigger*] [-c] [-y *update-buffer-sectors-count-for-deferred-write-trigger-event*]}}]	*Global buffers*
239	[pdhagroup -g *HA-group-identifier* -u *unit-identifier*[,*unit-identifier*]...]	*HA groups*
240	[pdstbegin[-k *statistical-information-type*[,*statistical-information-type*]...] [-m *interval*] [{-x *host-name* | -u *unit-identifier* }] [{ -a | -s *server-name*[, *server-name*]...}]] [-w]	*Statistical information*
241	[pdhibegin -k *statistics-type*[,*statistics-type*]...]	
242	{{[pdcltgrp -g *client-group-name* -u *guaranteed-number-of-connected-users-per-group*]}}	*Client group*
243	{{[pdplugin -n *plug-in-name*]}}	*Plug-ins*
244	**[putenv SHMMAX *maximum-shared-memory-segment-size*]**	*Shared memory*
245	[putenv TZ *time-zone*]	*Date and time*
246	[pdmlgput -s *output-selection* {-c ALL |[-l *message-severity*] -m *message-ID*[,*message-ID*]...}]	*Message output suppression facility*

2.2 Operand explanations

2.2.1 Operands related to the system structure

1) pd_system_id = *HiRDB-identifier*

~<identifier>((4 alphabetics))

Specifies an identifier for the HiRDB system. This operand is required.

If you are using multiple HiRDB/Single Servers, specify a unique HiRDB identifier in each system.

Notes

The only way to change a HiRDB identifier after it has been set is to use the database initialization utility to rebuild the system. For this reason, avoid specifying a name that might have to be changed later.

2) pd_name_port = *HiRDB-port-number*

~<unsigned integer> ((5001-65535)) <<20000>>

Specifies the port number of the HiRDB port.

Note

For details about HiRDB port numbers, see *Port numbers used by HiRDB* in the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand is related to the `-p` option of the `pdunit` operand.

3) pd_master_file_name = "*HiRDB-file-name-at-beginning-of-master-directory-RDAREA*"

~<path name>((up to 167 characters))

Specifies as an absolute path name the name of the HiRDB file that is at the beginning of the master directory RDAREA.

This operand is required.

Notes

- Once the name of the HiRDB file that is at the beginning of the master directory RDAREA is specified, it cannot be changed. To change it, you must rebuild the system with the database initialization utility. For this reason, avoid specifing a name that might have to be changed later.
- HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\rdarea\mast\mast01, C:\rdarea\mast` is not case sensitive, but `mast01` is case sensitive.

2.2.2 Operands related to maximum concurrent executions

4) pd_max_users = *maximum-number-of-concurrent-connections*

- For a HiRDB/Single Server: ((1-3000))<<10>>
- For a HiRDB/Parallel Server: ((1-2000))<<10>>

This operand specifies the maximum number of concurrent connections allowed for the HiRDB server. For a HiRDB/Parallel Server, this operand specifies the maximum number of concurrent connections allowed for a single front-end server. If the number of connection requests exceeds the value specified by this operand, HiRDB rejects any additional connection requests (connection requests result in errors). Note that connection in this case means an attempt to connect to a HiRDB server based on a CONNECT statement.

Specification guidelines

- When OpenTP1 is used, the connection count is the number of OpenTP1 server processes to be connected to the HiRDB server (including XA connections).
- When DBPARTNER/Server is used, the connection count is the number of DBPARTNER/Server connection clients.
- When HiRDB SQL Executer is used, the connection count is the number of HiRDB SQL Executer connection clients.

- If the multi-connection facility is used, the connection count is the total number of concurrent connections by individual UAPs.
- Some of the HiRDB commands and utilities are internally connected to the HiRDB server. Therefore, while these commands and utilities are being executed, the number of users that can be connected temporarily decreases. Determine the operand value by taking this fact into consideration. For details about the number of command and utility connections, see *Number of Concurrent Command Connections* in the manual *HiRDB Version 9 Command Reference*.

Notes

- If you increase the value of the `pd_max_users` operand, you must also increase the value of the `pd_max_server_process` operand. Along with these increases, both the size of the shared memory and the number of ports used by the HiRDB system also increase. For notes related to an increase in the value of the `pd_max_users` operand (when the number of users is increased), see the *HiRDB Version 9 System Operation Guide*. If you omit the `pd_max_server_process` operand, HiRDB automatically computes a value for this operand.
- For a multiple front-end server, the maximum number of connection requests that can be processed is obtained by multiplying the value of the `pd_max_users` operand by the number of front-end servers. However, make sure that the value obtained does not exceed the specifiable upper limit for the `pd_max_users` operand (`2000`).
- Connection requests to the HiRDB server in excess of this operand's value will result in an error, and processes might be left over. Use the `pdls -d prc` command to check for leftover processes; if any are found, terminate them forcibly with the `pdcancel` command.

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_hash_entry`
- `pd_lck_pool_size`
- `pd_lck_release_detect`
- `pd_max_bes_process`
- `pd_max_dic_process`
- `pd_max_server_process`
- `pd_process_count`
- `pd_process_terminator_max`
- `pdcltgrp`

Effects on individual estimation formulas

If the value of the `pd_max_users` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Considerations when migrating to 64-bit mode*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Processes started by HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Size of a work table file used by an SQL statement*
- *Determining the maximum number of files (pdfmkfs -l command)*
- *Maximum and minimum values for the system configuration*
- *Determining the number of records in a synchronization point dump file*
- *Formula 1*, *Formula 2*, *Formula 3*, and *Formula 4* under *Formulas for shared memory used by a single server*

- *Formula 1* and *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 1*, *Formula 2*, *Formula 3*, and *Formula 4* under *Formulas for the size of the shared memory used by a back-end server*
- *Formula for the shared memory used by a front-end server*
- *Estimating the number of ports that a unit will use*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

5) pd_max_server_process = *maximum-number-of-concurrently-activated-server-processes*

~<unsigned integer>((50-10000))

Specifies the maximum number of processes that can be activated at the same time within a single unit. The number of server processes includes the number of processes for the system server, individual servers, utilities, and the like (the system server is a server that is used internally by HiRDB).

Specification guidelines

- Normally, omit this operand. When it is omitted, the value obtained from the following formula is assumed. HiRDB automatically re-computes the value of this operand if you change any of the operands mentioned in the explanation of a variable.

  Default value = $a + b \times (c + 27) + 73 + i + j + k + n + o$

  When `v6compatible` is specified in the `pd_sysdef_default_option` operand, the default for this operand is `100`.
- If you choose to specify this operand, use the following formula as a reference for determining the operand value. For a HiRDB/Parallel Server, determine values for all units, and use the largest of these values as a guideline.

Recommended value = $a + b \times (c + d \times e + f) + d \times g + h + i + j + k + n + o + A$

Variable	Explanation of variable
a	**For HiRDB/Single Server:** Value of `pd_max_users` **For HiRDB/Parallel Server:** Total the values obtained using the following formulas for individual servers inside the unit: Back-end server: Value of `pd_max_bes_process` Dictionary server: Value of `pd_max_dic_process` Front-end server: Value of `pd_max_users` • If the `pd_max_bes_process` or `pd_max_dic_process` operand is omitted, use the value of the `pd_max_users` operand. • If a unit contains multiple back-end servers, compute a value for each back-end server. • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, substitute the value of the `pd_ha_max_server_process` operand.
b	**For HiRDB/Single Server:** 1 **For HiRDB/Parallel Server:** Server count inside the unit (server count allocated to the unit by the `pdstart` operand of the system common definition) • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, add the value of the `pd_ha_max_act_guest_servers` operand (or the default, if this operand has been omitted).
c	Concurrently executable process count in full recovery processing (value of the `pd_max_recover_process` operand)
d	Maximum number of concurrently executable utilities ("2" is the default value for this operand)
e	Number of processes to be started in each server by the utility (set this variable to 10)

Variable	Explanation of variable
f	Number of processes to be started by HiRDB for controlling the server (set this variable to 6)
g	Number of processes to be started in each unit by the utility (set this variable to 10)
h	Number of processes to be started by HiRDB for controlling the unit (set this variable to 50)
i	**For HiRDB/Single Server:** 1 **For HiRDB/Parallel Server:** Number of back-end servers in the unit • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, substitute the value of the `pd_ha_max_act_guest_servers` operand.
j	**For HiRDB/Single Server:** value of the `pd_max_ard_process` operand **For HiRDB/Parallel Server:** *number of back-end servers in the unit* × value specified for the `pd_max_ard_process` operand + *number of dictionary servers in the unit* × value specified for the `pd_max_ard_process` operand • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, add the following value: *maximum value of pd_max_ard_process operand for guest BES* × *value of the pd_ha_max_act_guest_servers operand*
k	Depends on the value specified for the `pd_process_terminator` operand. • `resident`: 1 • `fixed` (default value): `pd_process_terminator_max` operand value • `nonresident`: 0
n	**For HiRDB/Single Server:** `pd_dfw_awt_process` operand value **For HiRDB/Parallel Server:** *number of back-end servers* × *pd_dfw_awt_process operand value* + *number of dictionary servers* × *pd_dfw_awt_process operand value* • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, add the following value: *maximum value of pd_dfw_awt_process operand value for guest BES* × *pd_ha_max_act_guest_servers operand value*
o	When using a memory database, substitute the following value: *p* + 128 + *XDS count in the unit* × 5 + 2 **If the unit has a system manager** *p*: 64 × *XDS count for the system as a whole* **If the unit does not have a system manager** *p*: 0
A	Maximum number of concurrently activated transaction recovery processes $\downarrow$ *pd_trn_rcvmsg_store_buflen operand value* ÷ 72 $\downarrow$

Relationship to other operands

- This operand's value limits the maximum value of the `pd_process_count` operand.
- If you need to adjust the value for each unit because the server configuration in each unit is different, specify the `pd_max_server_process` operand in the unit control information definition.

Notes

- This specification value includes the number of processes for servers and utilities in the unit. If this value is too small, the following might occur:
  - Unit or server startup process results in an error
  - Transaction recovery cannot be performed
  - HiRDB planned termination cannot be performed
  - If the number of concurrent connections becomes greater than the maximum number of resident processes, connections can no longer be made
  - A command error results when an internal process activated by an extension of a command's execution starts.
- Because the number of processes that can actually be activated depends on factors such as the system resources, it might be necessary to adjust the resources or to change the locations of servers in some cases.

Effects on individual estimation formulas

If the value of the `pd_max_server_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Estimating desktop heap settings*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

6) pd_max_access_tables = *concurrently-accessible-base-tables-count*

~<unsigned integer>((4-32000))<<64>>

Specifies the maximum combined number of tables and sequence generators that can be accessed concurrently in a single transaction. This refers to the number of tables plus the number sequence generators coded in the SQL statements in a transaction.

Specification guidelines

- If the same table or sequence generator is specified in different SQL statements, the count must include each time the table or sequence generator is used.
- If the same table or sequence generator is specified more than once in the same SQL statement, the count must include each time the table or sequence generator is used.
- To set up referencing privileges for dictionary tables, the value obtained by multiplying the number of dictionary tables to be accessed by 5 must be added to the concurrently accessible base tables count (for details about the dictionary table referencing privilege, see the *HiRDB Version 9 System Operation Guide*).
- Specify a value greater than the number of times the `load` statement is specified in the HiRDB Datareplicator import table definition. Specifying a value smaller than the number of `load` statements will result in the `KFPA11931-E` error.
- HiRDB manages individually for each HiRDB server process the entries in tables that can lock a transaction based on the value specified in this operand. If the value is insufficient, an SQL error will result.

Relationship to other operands

When `v6compatible` is specified in the `pd_sysdef_default_option` operand, the default for this operand is `16`.

Effects on individual estimation formulas

If the value of the `pd_max_access_tables` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Considerations when migrating to 64-bit mode*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula 1* and *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 1*, *Formula 2*, and *Formula 3* under *Formulas for the size of the shared memory used by a back-end server*
- *Formula for the shared memory used by a front-end server*

7) pd_utl_exec_mode = <u>0</u> | 1

Specifies whether to permit increases in the maximum number of concurrent executions of utilities.

`0`:

Do not permit increases in the maximum number of concurrent executions of utilities. When `0` is specified, the HiRDB system determines a fixed maximum number of concurrent executions of utilities.

`1`:

Permit increases in the maximum number of concurrent executions of utilities. When `1` is specified, the maximum number of concurrent executions of utilities can be increased on the basis of the value of the `pd_max_users` operand.

For details about the maximum number of concurrent executions of utilities that is determined by the value specified in this operand, see *Maximum number of concurrently executable utilities* in the manual *HiRDB Version 9 Command Reference*.

Specification guidelines

Normally, `0` is specified, but specify `1` in the following cases:

- When the maximum number of concurrent executions with `0` specified is insufficient.
- When the following formula applies:

total back-end server count (1 for a HiRDB/Single Server) $\times$ 2 $\times$ *concurrent number of executions for all utilities* $\geq$ 824

In addition, when both of the following conditions apply, specifying `1` in this operand can reduce the amount of shared memory used.

- *value of pd_max_users* < 32
- *value of pd_max_users* > *maximum-number-of-concurrent-executions-of-utilities*

When `1` is specified, the shared memory needed for utility execution is allocated dynamically during utility execution.

Effects on individual estimation formulas

If the value of the `pd_utl_exec_mode` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

8) pd_max_commit_write_reclaim_no = *maximum-number-of-concurrent-executions-of-pdreclaim-commands-with-p-option-specified*

- For a HiRDB/Single Server: ((0-3000))<<10>>
- For a HiRDB/Parallel Server: ((0-2000))<<10>>

Specifies the maximum number of `pdreclaim` commands with the `-p` option specified that can be executed concurrently. For a HiRDB/Parallel Server, this operand specifies the maximum number of concurrent executions per server.

Specifying 0 results in termination with an error of any `pdreclaim` command in which the `-p` option is specified. When the number of concurrent executions of `pdreclaim` commands in which the `-p` option is specified exceeds the value of this operand, the excess `pdreclaim` commands in which the `-p` option is specified terminate with an error.

Specification guidelines

If the `-p` option is specified in the `pdreclaim` command, specify a value that adds a margin for the number of concurrent executions for the server that executes those commands. If you always omit the `-p` option from the `pdreclaim` command, specify 0 (to reduce the size of shared memory used).

Notes

If you specify a large value for this operand, check and, if necessary, revise the memory requirement, because the shared memory used by the back-end server, dictionary server, and single server increases. For details about estimating the shared memory, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, 0 is assumed when this operand is omitted. This means that if you specify `v6compatible` or `v7compatible` in the `pd_sysdef_default_option` operand and omit this operand, `pdreclaim` commands in which the `-p` option is specified cannot be executed.

Effects on individual estimation formulas

If the value of the `pd_max_commit_write_reclaim_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 5* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 6* under *Formulas for the size of the shared memory used by a back-end server*

2.2.3 Operands related to HiRDB startup

9) pd_mode_conf = AUTO | MANUAL1 | <u>MANUAL2</u>

Specifies the HiRDB startup method.

`AUTO`:

HiRDB is automatically started. The automatic startup method automatically starts HiRDB when the OS is started.

However, you must manually start HiRDB in the following cases. In the manual startup method, you must execute the `pdstart` command to start HiRDB.

- Normal start following normal termination (without restarting the OS)
- Restart following planned termination (without restarting the OS)
- Restart following forced termination

`MANUAL1`:

HiRDB is manually started. However, after abnormal termination, HiRDB is automatically started.

`MANUAL2`:

Manual startup.

Specification guidelines

- To restart HiRDB (or the unit) automatically after HiRDB (or the unit) has terminated abnormally, specify `AUTO` or `MANUAL1`.

- If you use the system switchover facility, specify `MANUAL1` or `MANUAL2`. Which of these modes is more appropriate depends on how the system switchover facility is used. For details about how to operate the system switchover facility, see the *HiRDB Version 9 System Operation Guide*.
- The startup mode is determined by the combination of the previous termination mode (normal termination, forced termination, planned termination, or abnormal termination) and the startup method specified by this operand, as shown in the following table:

pd_mode_conf specification	Previous termination mode	Startup mode	Startup method
`AUTO`	Normal termination	Normal startup	Automatic startup or Manual startup[#1]
	Planned termination	Restart[#2]	
	Forced termination	Restart[#2]	Manual startup
	Abnormal termination	Restart	Automatic startup
`MANUAL1`	Normal termination	Normal startup	Manual startup
	Planned termination	Restart[#2]	
	Forced termination	Restart[#2]	
	Abnormal termination	Restart	Automatic startup
`MANUAL2`	Normal termination	Normal startup	Manual startup
	Planned termination	Restart[#2]	
	Forced termination	Restart[#2]	
	Abnormal termination	Restart[#2]	

#1: Automatic startup is used only when the OS is started. For normal start following normal termination and restart following planned termination (without restarting the OS), manual startup is used.

#2: The `dbdestroy` option of the `pdstart` command can be used for a forced startup. However, when forced startup is used, the HiRDB database is not recovered automatically and must be recovered by the HiRDB administrator.

Note

- To use automatic startup (`pd_mode_conf=AUTO`) in a HiRDB/Parallel Server, try to start all units within 20 minutes after starting the first unit. If all units are not started within 20 minutes, the HiRDB startup process is cancelled. You can use the `pd_reduced_check_time` operand to change this 20-minute time limit.

  Note that this time limit does not apply when you are restarting a unit after it or the OS was abnormally terminated.

10) pd_system_complete_wait_time = ***pdstart-command-completion-wait-time***

~<unsigned integer>((610-3600))<<610>> (seconds)

Specifies the amount of time to wait for completion of the `pdstart` command's processing. This operand needs to be specified only when it is necessary to provide a longer wait time than the default value of 610 seconds (10 minutes and 10 seconds). If startup processing is not completed within the specified number of seconds after the `pdstart` command was entered, HiRDB outputs the `KFPS05078-I` message and returns the `pdstart` command with an error.

Even though the `pdstart` command is returned with an error, startup processing continues and HiRDB is started. Therefore, specify this operand only in the following cases:

- When the `KFPS05078-I` message is the monitoring target
- When another operation is to be performed automatically after the `pdstart` command terminates normally

Application criterion

Normally, this operand need not be specified. This operand is applied when the `KFPS05078-I` message is output and the `pdstart` command is returned with an error during HiRDB startup processing.

11) pd_start_time_out = ***HiRDB*****-start-preparation-maximum-wait-time**

~<unsigned integer>((1-1440))<<15>>(minutes)

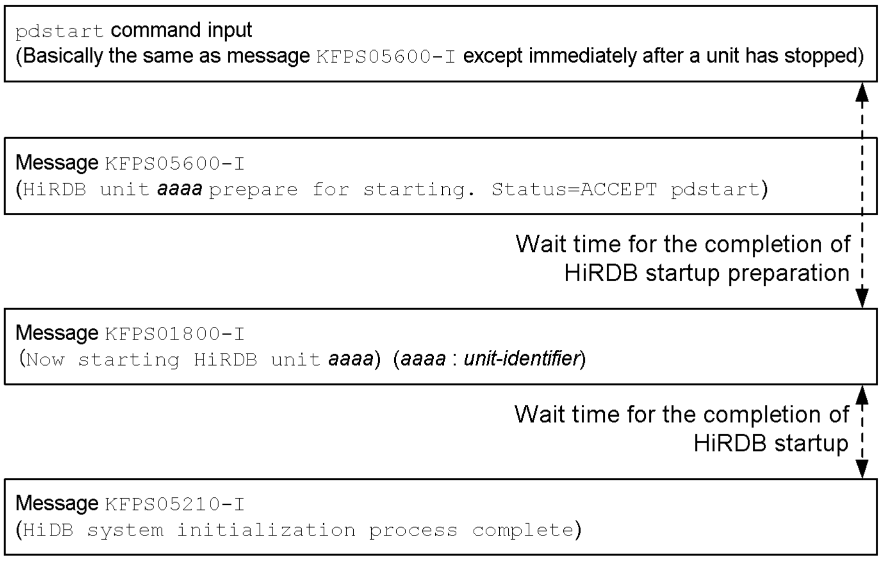
Specifies the maximum amount of time to wait for completion of HiRDB start preparation after `pdstart` command input. If HiRDB start preparation is not completed within the time specified in this operand after `pdstart` command input, the `KFPS01861-E` message (`reason code = TIMEOUT`) is output in the `pdstart` command input window.

Specification guidelines

Normally, you need not specify this operand. In an ordinary environment, the HiRDB start preparation time is at most 3 to 6 minutes, and therefore the default value [15 minutes] of this operand is sufficient for the maximum wait time. In rare cases, HiRDB start preparation might take more than 15 minutes because of factors such as server machine performance, disk size, and memory load. In this case, the `KFPS01861-E` message (`reason code = TIMEOUT`) is output in the `pdstart` command input window. If this occurs, specify a value greater than 15 minutes for this operand.

Reference note

Based on the time at which the following message was output, you can compute the wait time for the completion of the HiRDB startup preparation process and the wait time for the completion of the HiRDB startup:


`pdstart` command input
(Basically the same as message `KFPS05600-I` except immediately after a unit has stopped)

Message `KFPS05600-I`
(`HiRDB unit` ***aaaa*** `prepare for starting. Status=ACCEPT pdstart`)

Message `KFPS01800-I`
(`Now starting HiRDB unit` ***aaaa***) (***aaaa*** : *unit-identifier*)

Message `KFPS05210-I`
(`HiDB system initialization process complete`)

Wait time for the completion of HiRDB startup preparation
Wait time for the completion of HiRDB startup

2.2.4 Operands related to reduced activation

12) pd_start_level = 0 | 1

This operand is applicable only to HiRDB/Parallel Server.

Specifies whether reduced activation is to be used when a unit cannot be started during HiRDB startup. For details about reduced activation, see the *HiRDB Version 9 System Operation Guide*.

`0`:

Do not use reduced activation. HiRDB will not start if an error prevents any unit from being activated.

`1`:

Use reduced activation. Even if an error prevents a unit from being activated, HiRDB (all other units) will be started.

13) pd_reduced_check_time = ***wait-time-for-reduced-activation-startup-notice***

~<unsigned integer>((300-1200))<<1200>>(seconds)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the amount of time to wait for receipt of a notice of activation completion from each unit. If the activation completion notice is not received from a unit before the amount of time specified here elapses, reduced activation without that unit is used.

Condition

1 must be specified for the pd_start_level operand.

Specification guidelines

Before specifying this operand, determine how long it takes for the KFPS01826-I message to be output after the pdstart command is entered (the KFPS01826-I message is output to the unit that defines the system manager). The value specified here must be greater than this message's output time.

Notes

Even when the wait time specified by this operand elapses, reduced activation cannot be executed if all the conditions for reduced activation have not been satisfied. In this case, HiRDB is abnormally terminated. For details about the conditions for reduced activation, see the *HiRDB Version 9 System Operation Guide*.

14) pd_start_skip_unit = *name-of-unit-not-to-be-started*[*,name-of-unit-not-to-be-started*]...

This operand is applicable only to HiRDB/Parallel Server.

Specifies the names of units that need not be started when reduced activation is executed. The units specified here are excluded from the startup process.

Condition

The pd_start_level operand must be set to 1.

Advantage

During a startup, HiRDB waits for up to 20 minutes for activation startup notifications from all units. If reduced activation is in effect, HiRDB will wait the full 20 minutes for units that cannot be started. If units that cannot be started are specified in this operand, HiRDB will not wait for activation startup notifications from them, thus saving up to 20 minutes of wait time and reducing the time for reduced activation to be implemented.

Note

- If the standby-less system switchover (effects distributed) facility is being used, not all units within the HA group need to be active, and therefore specification of this operand is invalid. However, all back-end servers within the HA group must be running at some unit for HiRDB to be able to start.
- Even if a recovery-unnecessary front-end server unit is specified for this operand, that server unit is not treated as a reduced activation target.

2.2.5 Operands related to HiRDB processing

15) pd_dbsync_point = sync | commit

Specifies the timing for committing database updates to the file.

sync:

Commit database updates to the file at each synchronization point. This option enhances performance when a large number of transactions that update the same page occur between synchronization points. Because update information is not committed to the file when a COMMIT statement is issued, the input/output workload is reduced. Note that full recovery processing is slower than when commit is specified.

commit:

Commit database updates to the file when a COMMIT statement is issued. Because the database contents are guaranteed when the transaction is completed, there is no need to recover transaction processing from a synchronization point, thus reducing the time required for a full recovery. However, if a large number of transactions that update the same page occur between synchronization points, this option is slower than when the sync option is specified.

Remarks

A LOB RDAREA is not affected by this operand. Directories are updated when the COMMIT statement is issued. Whether data is updated depends on whether a LOB global buffer has been allocated. If no LOB global buffer has been allocated, data is instantly updated when an update request is issued. If a LOB global buffer has been allocated, data is updated when the COMMIT statement is issued. However, if the global buffer becomes full, data is updated at that time.

Relationship to other operands

This operand is related to the pd_system_dbsync_point operand.

Effects on individual estimation formulas

If the value of the `pd_dbsync_point` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a single server*
- *Formulas for the size of the shared memory used by a dictionary server*
- *Formulas for the size of the shared memory used by a back-end server*

16) pd_system_dbsync_point = sync | <u>commit</u>

Specifies the timing for committing to file updates in the following types of RDAREAs:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs
- Data dictionary LOB RDAREAs
- Registry RDAREAs
- Registry LOB RDAREAs

`sync`:

Commit RDAREA updates to the file at each synchronization point. Because update information is not committed to the file when a `COMMIT` statement is issued, the processing performance of a definition SQL is slightly better than when the `commit` option is specified. However, full recovery processing is slower than when the `commit` option is specified.

`commit`:

Commit to file updates to the indicated types of RDAREAs when a `COMMIT` statement is issued. Because the contents of updates to the indicated types of RDAREAs are guaranteed when the transaction is completed, there is no need to recover these types of RDAREAs from a synchronization point, thus reducing the time required for a full recovery. However, the processing performance of a definition SQL is slightly lower than when `sync` is specified.

Relationship to other operands

This operand is related to the `pd_dbsync_point` operand. The following table shows the relationship to the `pd_dbsync_point` operand:

pd_dbsync_point specification	pd_system_dbsync_point specification	
	sync	commit (default)
`sync` (default)	Commits updates to all RDAREAs at a synchronization point.	Commits updates to the indicated types of RDAREAs when a `COMMIT` statement is issued. Commits updates to other types of RDAREAs at a synchronization point.
`commit`	Commits updates to all RDAREAs when a `COMMIT` statement is issued.	

Effects on individual estimation formulas

If the value of the `pd_system_dbsync_point` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a single server*
- *Formulas for the size of the shared memory used by a dictionary server*

17) pd_dbsync_altwrite_skip = Y | <u>N</u>

Specifies the handling of write processing when an update buffer reference request is issued during synchronization point acquisition processing. Normally, such write processing of the update buffer contents into the database is handled by the server process that will execute the transaction that issues the reference request. This operand specifies whether the handling of this write processing is to be skipped.

The following compares the values of this operand.

Comparison item	pd_dbsync_altwrite_skip operand value	
	Y	N (default value)
Database writing method when an update buffer reference request is issued during synchronization point acquisition processing	The server process that will execute the transaction that issues the reference request does not handle writing of the update buffer contents into the database. (The handling is skipped.)	The server process that will execute the transaction that issues the reference request handles writing of the update buffer contents into the database. (The handling is not skipped.)
Advantage	Because the server process that will execute the transaction does not handle the write processing, the performance of the referencing transaction is stable during synchronization point acquisition processing.	The amount of time required for synchronization point acquisition processing is reduced because some of the workload is distributed to the referencing transaction.
Disadvantage	The amount of time required for synchronization point acquisition processing increases because none of the workload is distributed to the referencing transaction.	There might be adverse effects on the referencing transaction during synchronization point acquisition processing.

Specification guidelines

You can execute the statistics analysis utility to obtain statistical information on the global buffer pool in order to check the database write processing take-over count (`ALTRW`) due to reference request hits during synchronization point processing. If this value is large, performance of a referencing transaction is not stable during synchronization point acquisition processing. To achieve stable performance, specify `Y`. Note that when `Y` is specified, more time is required for synchronization point acquisition processing; if necessary, you can use the facility for parallel writes in deferred write processing to distribute the write processing workload.

18) pd_process_terminator = resident | <u>fixed</u> | nonresident

If a HiRDB process is abnormally terminated, HiRDB starts a process that executes *post-processing*. This operand specifies whether the post-processing process is activated when HiRDB is started.

`resident`:

This option starts a single post-processing process when starting HiRDB. For a HiRDB/Parallel Server, a post-processing process is started in each unit.

If multiple processes terminate abnormally at the same time, post-processing processes up to the number specified by HiRDB are started and executed in parallel. If a new post-processing process cannot be started due to memory shortage, for example, post-processing is sequentially performed using the post-processing processes that are already active.

If the number of post-processing processes specified by HiRDB cannot be started due to a memory shortage, HiRDB (or the affected unit for a HiRDB/Parallel Server) might terminate abnormally.

`fixed`:

When starting HiRDB, this option starts the number of post-processing processes specified by the `pd_process_terminator_max` operand. For a HiRDB/Parallel Server, the number of post-processing processes specified by the `pd_process_terminator_max` operand are started in each unit. If post-processing processes cannot be started due to memory shortage, for example, HiRDB (or the applicable unit for a HiRDB/Parallel Server) is not started.

If a number of processes exceeding the value specified in the `pd_process_terminator_max` operand terminate abnormally at the same time, no additional post-processing processes are started. In this case, post-processing is sequentially performed using the post-processing processes that are already active.

`nonresident`:

This option does not start any post-processing process when starting HiRDB. A post-processing process is started whenever a process terminates abnormally.

If multiple processes terminate abnormally at the same time, post-processing processes are simultaneously started and executed in parallel. If post-processing processes cannot be started due to memory shortage, for example, HiRDB (or the applicable unit for a HiRDB/Parallel Server) might terminate abnormally in some cases.

Specification guidelines

- To improve reliability, specify `resident` or `fixed`. Although `fixed` provides higher post-processing performance than `resident`, `fixed` requires more memory.
- When `nonresident` is specified, post-processing processes are started on demand. Consequently, post-processing processes cannot be started if memory shortage occurs. Furthermore, if multiple processes terminate abnormally at the same time, multiple post-processing processes are started, resulting in performance degradation.
- If you want to prevent HiRDB from terminating abnormally when a post-processing process cannot be started, we recommend that you specify `fixed`.

Note

You must be careful when changing the specification value to `fixed`. Because this option starts post-processing processes when starting HiRDB, it requires more memory. If memory shortage, for example, prevents post-processing processes from being started, HiRDB (or the applicable unit for a HiRDB/Parallel Server) cannot start.

19) pd_process_terminator_max = *maximum-number-of-resident-post-processing-processes*

~<unsigned integer>((1-100)) << (3, ↑ value of `pd_max_users` ÷ 100 **↑**)>>

Specify this operand if you have omitted the `pd_process_terminator` operand or specified `fixed` or it. Specify for the `pd_process_terminator_max` operand the number of post-processing processes to be started when starting HiRDB. If memory shortage, for example, prevents the specified number of post-processing processes from being started, HiRDB (or the applicable unit for a HiRDB/Parallel Server) cannot start.

Specification guidelines

The number of post-processing processes needed is proportional to the value of `pd_max_users`. A small value might delay recovery processing, while a large value might use up memory unnecessarily.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is as follows:

↑ (value of `pd_max_users` ÷ 100 ↑

20) pd_process_desktopheap_size = *size-of-desktop-heap-used-per-process*

~<unsigned integer>((50-10000))(bytes)

- Windows XP, Windows Vista, Windows Server 2008, Windows Server 2012, Windows 7, or Windows 8: <<`5000`>>
- Windows Server 2003: <<`100`>>

Specifies the size of the desktop heap used per process in HiRDB.

Specification guidelines

- Normally, there is no need to specify this operand.
- Use the following formula to determine the desktop heap size for HiRDB:

  Desktop heap size for HiRDB = number of desktops created by HiRDB × *desktop heap size per desktop*[#1] (bytes)

  Number of desktops created by HiRDB = ↑ (*pd_max_server_process operand value* + 50) ÷ *desktop heap size per desktop*[#1] × *pd_process_desktopheap_size operand value* ↑

  The maximum desktop heap size is 48 megabyes[#2] for the entire server machine. If the total size of the desktop heap, including other processes that are running, exceeds 48 megabytes[#2], take the following actions:

  ● Determine the desktop heap size for HiRDB, reduce the `pd_max_server_process` operand value, and then restart HiRDB.

  ● Terminate other processes that are running.

  #1: This is the size of the non-interactive desktop heap specified in the OS registry.

  #2: This is the maximum value for the default desktop heap for Windows Server 2003 (32 bits). The maximum value for the desktop heap depends on the OS.

21) pd_server_entry_queue = <u>spnfifo</u> | fifo | loop

If contention occurs in the HiRDB server process during concurrent execution of UAPs, processing requests might sometimes be temporarily queued. This operand specifies what HiRDB must do in this case. Note that *process contention* in this case means that multiple processes are simultaneously trying to lock internal resources, such as tables and RDAREAs, when transactions are running on the HiRDB server process. Only a single process is allowed to lock internal resources at any point in time. *Spin*, referred to in the following explanation means a process for acquiring the right to execute a lock. When another process releases the right to execute a lock, a process that is spinning has a higher probability of acquiring the right to execute a lock.

`spnfifo`:

A processing request that occurs first is given higher priority. However, because the process is spun a certain number of times before being registered in a queue, the priority order is not perfect. This method is used in Version 06-00 and earlier versions.

`fifo`:

A processing request that occurs first is given higher priority than when `spnfifo` is specified. Because no spinning occurs before a process is registered in a queue, the priority order is maintained better than when `spnfifo` is specified. This method also reduces the CPU load.

`loop`:

All processing requests are given the same priority. When processes are registered in a queue, they are spun at high speed. Specifying `loop` might improve the response during concurrent execution of UAPs. However, this method places a greater load on the CPU than other methods.

Specification guidelines

Normally, you need not specify this operand.

Change the specification value if the processing performance during concurrent execution of UAPs does not improve. Doing so might improve the performance.

22) pd_thdlock_sleep_func = <u>0</u> | 1

Specifies the process standby method to be used when acquiring a lock on shared resources, such as shared memory. The following table shows the relationship between the values specified in this operand and in the `pd_thdlock_retry_time` operand.

pd_thdlock_sleep_func operand value	pd_thdlock_retry_time operand value	
	1 to 10000	10001 to 1000000
`0`	The processes go on standby for only the amount of sleep time set for the inter-thread lock specified with `select()` or `Sleep()`.	
`1`	The OS determines process allocation using `sched_yield()` or `SwitchToThread()` (the `pd_thdlock_retry_time` operand value is ignored)#	The processes go on standby for only the amount of sleep time set for the thread lock specified with `select()` or `Sleep()`.

#: Because processes do not go on standby, CPU usage increases.

Specification guidelines

- Specification of this operand is optional. Either specify `0` or omit this operand.

Notes

- If you change the value of this operand from `0` to `1`, CPU usage might increase, resulting in performance degradation in some cases.
- Processing to obtain a thread lock might not always work on the first try. Retry until a lock is acquired.
  If you retry with `1` specified in this operand, no sleep processing will result, so if CPU usage rises to 100%, the overhead associated with obtaining the thread lock will become a major issue during multiplexed execution of processing. This means that even when the equivalent number of transactions are inserted prior to CPU usage reaching 100%, CPU usage might stay at 100% for an extended period.

Relationship to other operands

Specification of this operand is invalid if a value greater than `10000` is specified for the `pd_thdlock_retry_time` operand.

23) pd_thdlock_wakeup_lock = Y | N

Specifies a thread lock release notification method. Specify `Y` in this operand to ensure that release notifications are transmitted.

`Y`:

When issuing a thread lock release notification, a new separate lock is temporarily obtained.

`N`:

When issuing a thread lock release notification, no new separate lock is temporarily obtained.

This is the HiRDB processing mode for versions 06-02 and earlier.

Specification guidelines

Specify `Y` in this operand.

Notes

Note the following about specifying `N` or omitting this operand:

- Transactions might occur that have longer execution times than for other transactions, reducing response during multiplexed UAP execution.
- When no notification is sent that a thread lock has been released, there might be a delay equal to the amount of time specified in `pd_thdlock_pipe_retry_interval` in obtaining the lock that is waiting for release.

24) pd_thdlock_pipe_retry_interval = *thread-lock-release-check-interval*
~<unsigned integer>((0-2147483647))<<1000000>>(microseconds)

Specifies the interval in microseconds at which to check for thread lock release.

Specification guidelines

If the value that is set is the same as or greater than the default value and all the following conditions are satisfied, CPU usage might decrease:

- `pd_thdlock_wakeup_lock` = `Y` is specified.
- Processing performance does not increase during multiplexed UAP execution.
- The CPU usage rate is very high.

However, transactions tend to occur that have longer execution times than for other transactions.

Do not specify this operand when the conditions above are not applicable.

Note

If a value less than the default value is specified, release checking is repeated at short intervals, which might the increase the CPU usage rate.

25) pd_thdlock_retry_time = *thread-lock-sleep-time*
~<unsigned integer>((1-1000000))<<10000>>(microseconds)

Specifies the thread lock sleep time in microseconds. If this operand is specified when all the conditions listed below are satisfied, the CPU usage rate might decrease:

- The CPU usage rate is very high.
- Reducing the CPU usage rate is necessary, even if it results in a reduction in performance.
- `0` is specified for the `pd_thdlock_sleep_func` operand.

Do not specify this operand when the conditions above are not applicable.

The following describes the HiRDB processing based on the combination of the `pd_thdlock_sleep_func` and the `pd_thdlock_retry_time` operand values:

pd_thdlock_sleep_func operand value	pd_thdlock_retry_time operand value	
	1 to 10000	10001 to 1000000
`0`	Each process stands by for the thread lock sleep time specified by `select()` or `Sleep()`.	
`1`	The OS determines process allocation using `sched_yield()` or `SwitchToThread()` (the `pd_thdlock_retry_time` operand value is ignored).#	Each process stands by for the thread lock sleep time specified by `select()` or `Sleep()`.

#: Because processes do not go on standby, CPU usage increases.

Specification guidelines

If you specify this operand, start by specifying `10000`. If the CPU usage rate is still too high, increase the value.

Notes

- Reducing the value might not change the performance.
- Specifying `1000` or a greater value might have an adverse effect on performance.

Relationship to other operands

If `v6compatible` is `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `10000`.

26) pd_thdspnlk_spn_count = *thread-spin-lock-spin-count*
~<unsigned integer>((0-2147483647))<<512>>

Specifies a spin count for thread spin lock. Specifying this operand when all of the following conditions are satisfied might improve the system performance. Otherwise, there is no need to specify this operand.

- An ample margin exists in the CPU usage rate.
- You want to improve performance, even if doing so increases the CPU usage rate.

Specification guidelines

- If this operand is to be specified, specify a value that is greater than the default value (`512`).
- Because the specification value depends on the OS type, the processor type, the machine performance, the disk performance, the UAP content, and the number of UAPs concurrently being executed, there is no clear guideline. Determine an appropriate value by varying the specification value and measuring the performance.

Notes

- If the value of this operand is too large, the CPU usage rate might increase, causing problems such as slower OS operation. In this case, decrease the operand value.
- Increasing the value of this operand might not always improve performance.

27) pd_pageaccess_mode = <u>SNAPSHOT</u> | NORMAL

Specifies the page access mode to be used for database search.

`SNAPSHOT`:

Uses a snapshot mode for page access. When the global buffer is accessed for the first time, rows that match the search condition are copied to the process private memory. During the second search request, a search result is returned by referencing the process private memory. For details about the snapshot mode, see the *HiRDB Version 9 Installation and Design Guide*.

`NORMAL`:

Uses a normal mode for page access. The global buffer is accessed for each search request.

Specification guidelines

If facilities for improving performance, such as the rapid grouping facility, cannot be used, consider using the snapshot mode. In normal search-SQL, the global buffer is accessed roughly the same number of times as the number of rows that match the specified search condition. Consequently, if search-SQLs are concurrently executed, accesses to the global buffer become concentrated, and as a result, the expected performance might not be obtained. In this case, using the snapshot mode can reduce the number of accesses by search-SQLs to the global buffer, and thus might improve the performance. However, using the snapshot mode increases the size of the process private memory used by HiRDB. For details about how to compute the size of process private memory when using the snapshot mode, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

When `v6compatible` is specified in the `pd_sysdef_default_option` operand, the default for this operand is `NORMAL`.

Effects on individual estimation formulas

If the value of the `pd_pageaccess_mode` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Procedure for obtaining the size of the memory required when the snapshot method is used* under *Estimating the memory size required for a HiRDB/Single Server*
- *Procedure for obtaining the size of the memory required when the snapshot method is used* under *Estimating the memory size required for a HiRDB/Parallel Server*

28) pd_cmdhold_precheck = Y | N

Specifies whether to check for RDAREA hold before locking the RDAREA.

`Y`:

Checks for RDAREA hold before locking the RDAREA.

`N`:

Does not check for RDAREA hold before locking the RDAREA. Checking is performed after locking.

The following hold types are checked:

- Command hold
- Reference-possible hold
- Reference-possible backup hold

Specification guidelines

Normally, omit this operand or specify `Y`. The following table describes the differences between `Y` and `N` specifications.

Item	Y specified	N specified
Processing by HiRDB	During UAP or command# execution, HiRDB checks the hold status of all RDAREAs that might be accessed before locking the RDAREAs. For example, when accessing a table that is row-partitioned into RDAREAs 1 through 4, HiRDB checks the hold status of all four RDAREAs. However, when conditions are specified by key range partitioning or `FIX` hash partitioning, and the RDAREAs that might be accessed by the UAP are narrowed, no error results even when RDAREAs that cannot be accessed are on hold.	During UAP or command# execution, HiRDB first locks RDAREAs and then checks the hold status of all RDAREAs that might be accessed. For example, assume that a table that is row-partitioned into RDAREAs 1 through 4 is to be accessed. If the target RDAREAs are narrowed using an index and if RDAREA 1 is to be accessed, HiRDB checks the hold status of RDAREA 1 only. This mode is used in HiRDB version 5.0 and earlier.
When a UAP accesses an RDAREA that is on hold	Because a hold check is performed before locking the RDAREA, the fact that the RDAREA is on hold can be detected more quickly than when `N` is specified.	Because a hold check is performed after locking the RDAREA, the locked RDAREA might cause a timeout error (`KFPA11770-E`) if a UAP accesses the RDAREA that is on hold. Additionally, if the access target RDAREA is on hold because data is being loaded or because it is being reorganized, the UAP might cause a hold error (`KFPA11920-E`).
When using a non-row partitioning index to narrow the access target RDAREAs	You must be careful when a table is row-partitioned but the index is not. When using a non-row partitioning index to narrow the access target RDAREAs. a hold error (`KFPA11920-E`) occurs, even when a non-access target RDAREA is on hold. In the example given for processing by HiRDB, the UAP causes a hold error (`KFPA11920-E`) if any of RDAREAs 1 through 4 is on hold.	When using a non-row partitioning index to narrow the access target RDAREAs, the UAP or command can be executed even if a non-access target RDAREA is on hold. In the example given for processing by HiRDB, the UAP can be executed even if RDAREAs 2 through 4 are on hold.

#: Refers to UAPs and commands that cannot be executed if RDAREAs are on hold.

29) pd_db_io_error_action = dbhold | unitdown

Specifies the processing to be performed by HiRDB when an input/output error occurs in an RDAREA (excluding the master directory RDAREA). If an input/output error occurs in the master directory RDAREA, HiRDB (or a unit for a HiRDB/Parallel Server) always terminate abnormally regardless of the specification in this operand. For the actions to be taken when an RDAREA input/output error occurs, see the *HiRDB Version 9 System Operation Guide*.

An input/output error in this case refers to an error that occurs when a file manipulation attempt by HiRDB fails due to a cause that cannot be determined by HiRDB. When such an error occurs, `-1544` is output as the error code returned in response to a HiRDB file system access request.

`dbhold`:

When an input/output error occurs in an RDAREA, the RDAREA is placed in an error shutdown state.

`unitdown`:

If an input/output error occurs in an RDAREA, HiRDB (or a unit for a HiRDB/Parallel Server) terminate abnormally. However, if an input/output error occurs again following an abnormal termination, the RDAREA is placed in an error shutdown state. To enable the specification of `unitdown` again, take one of the following actions:

- Start HiRDB normally.
- Execute the system reconfiguration command (`pdchgconf` command).

Specification guidelines

To determine the specification value for this operand, see *Actions to be taken when an RDAREA input/output error occurs* in the *HiRDB Version 9 System Operation Guide*.

Notes

- HiRDB terminates abnormally if an input/output error occurs while `unitdown` is specified. Consequently, in the following cases, the processing target RDAREA might go onto error shutdown status:

  ● The UAP or utility is executing in the pre-update log acquisition mode or the no-log mode.

  ● The UAP or utility is being executed on a user LOB RDAREA that has been placed in the no-log mode by specification of `NO` in the `RECOVERY` operand of `CREATE TABLE`.

  If you use the facility for taking a unit down when a physical error is detected, avoid running these operations, if possible. If you need to run these operations, make a backup prior to running the UAP or utility in case recovery from an RDAREA error shutdown needs to be performed. For details about making back-ups, see the *HiRDB Version 9 System Operation Guide*.
- If an input/output error occurs during the startup or termination process, HiRDB does not terminate abnormally even if `unitdown` is specified.
- During recovery processing by the database recovery utility (`pdrstr`), HiRDB does not terminate abnormally even though `unitdown` is specified. In such a case, re-execute `pdrstr` to perform recovery.

Relationship to other operands

This operand is related to the following operands:

- `pd_mode_conf` operand
- `pd_db_access_error_action` operand
- `pd_db_hold_action` operand

If `unitdown` is specified in more than one of the `pd_db_io_error_action`, `pd_db_access_error_action`, and `pd_db_hold_action` operands, the operand value that takes effect is determined in the following order:

1. `pd_db_io_error_action` operand
2. `pd_db_access_error_action` operand
3. `pd_db_hold_action` operand

If more than one RDAREA input/output, file access, or physical error has occurred, determine the error that caused unitdown based on the above priority. In addition, see the message that is issued.

30) pd_connect_errmsg_hide = Y | <u>N</u>

Specifies whether to hide the error cause in the message that is output when a connection attempt fails.

`Y`: Hides the error cause when a connection attempt fails.

`N`: Does not hide the error cause when a connection attempt fails.

Depending on the value specified for this operand, the message that is output when a connection attempt fails might vary. The following table shows the details:

Error cause	Output message	
	pd_connect_errmsg _hide = Y	pd_connect_errmsg _hide = N (default value)
Invalid authorization identifier (the specified user does not exist)	`KFPA19632-E`	`KFPA11561-E`
Invalid password (the specified password is invalid)	`KFPA19632-E`	`KFPA11560-E`

31) pd_rpc_bind_loopback_address = Y |<u>N</u> | S

Specifies whether a loopback address is to be used for `bind()` when the receiving port is generated.

`Y`: Use a loopback address for `bind()`.

`N`: Do not use a loopback address for `bind()`.

`S`:

The processes that accept connection requests only from processes in the HiRDB server use a loopback address for `bind()`. The processes that accept connection requests from HiRDB clients (system manager process and scheduler process) do not use a loopback address for `bind()`.

Condition

If `Y` is specified in this operand, all the following conditions must be satisfied:

- The HiRDB system consists of only HiRDB/Single Server# or the system switchover facility that inherits IP addresses is used in monitor mode.
- A loopback address is specified in the `-x` option in the `pdunit` operand and for the `PDHOST` operand in the client environment definition.

#

The HiRDB system consisting of only HiRDB/Single Server means that the following condition is satisfied:

- Both the HiRDB client and the HiRDB server are installed on the same machine (the HiRDB client is not installed on a separate machine).

If `S` is specified in this operand, all the following conditions must be satisfied:

- This is HiRDB/Single Server.
- A port number for the scheduler process is specified in one of the following system definitions:
  - `pd_scd_port` operand
  - `pd_service_port` operand
  - `-s` option in the `pdunit` operand
- A loopback address is specified in the `-x` option in the `pdunit` operand.
- If the HiRDB client connects to the HiRDB server from a separate server machine, it uses the high-speed connection facility and the following operands are specified in the client environment definition for the HiRDB client:
  - `PDSERVICEGRP`
  - `PDSERVICEPORT`
  - `PDSRVTYPE=PC`

Specification guidelines

The items to be added to the Windows Firewall exception list depends on the value of this operand as shown below. If you specify `Y` or `S`, you can eliminate or reduce the port numbers and programs that will be used by HiRDB and, therefore, added to the Windows Firewall exception list.

Value of pd_rpc_bind_loopback_address operand	Port numbers and programs that will be used by HiRDB and need to be added to the Windows Firewall exception list
`Y`	None
`N`	All port numbers and programs that will be used by HiRDB
`S`	• If port numbers are registered

Value of pd_rpc_bind_loopback_address operand	Port numbers and programs that will be used by HiRDB and need to be added to the Windows Firewall exception list

	HiRDB's port number#1 and the scheduler process's port number#2
	• If program names are registered
	`%PDDIR%\lib\servers\pdrdmd.exe`
	`%PDDIR%\lib\servers\pdscdd.exe`

#1

This is the value specified in one of the following system definitions:

- `pd_name_port` operand
- `-p` option in the `pdunit` operand

#2

This is the value specified in one of the following system definitions:

- `pd_scd_port` operand
- `pd_service_port` operand
- `-s` option in the `pdunit` operand

For details about how to add items to the Windows Firewall exception list, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand is related to the following operands:

- `pd_name_port`
- `pd_scd_port`
- `pd_service_port`
- `pdunit -p`
- `pdunit -s`
- `pdunit -x`
- `PDHOST`
- `PDSERVICEGRP`
- `PDSERVICEPORT`
- `PDSRVTYPE=PC`

32) pd_cancel_down_msgchange = Y | N

Specifies whether the error messages output when a server process is forcibly terminated are to be changed.

`Y`:

Changes the error messages to warning messages. The facility for changing error messages is called the facility for changing a process down message when cancelling a transaction.

`N`:

Does not change the error messages.

The following shows the relationship between the value of this operand and the error messages that are output:

Condition	Messages that are output	
	When Y (default value) is specified	When N is specified
Server process is terminated forcibly for one of the following reasons:# • Intentional forced termination by the user • Forced termination caused by a timeout	• `KFPS01852-W` • `KFPO00115-W`	• `KFPS01820-E` • `KFPO00105-E`

Condition	Messages that are output	
	When Y (default value) is specified	When N is specified
• Forced termination due to a failure at the client		
Server process is terminated forcibly for some other reason	• KFPS01820-E • KFPO00105-E	
HiRDB cannot identify the cause of the forced termination of the server process		

#: There are other causes that change the messages. For details, see *Facility for changing the process-down message when a transaction is cancelled* in the *HiRDB Version 9 System Operation Guide*.

Advantages

By specifying `Y` in this operand, you change the messages that are displayed for identifying the cause of forced termination of a server process.

Remarks

When the `KFPS01820-E` and `KFPO00105-E` messages are displayed, it is not possible to use the message IDs to distinguish between errors detected by HiRDB and errors resulting from an intentional user operation. To identify the cause, you must compare the process IDs that are displayed in the individual messages.

If JP1 is used to monitor messages, handling based on the `KFPS01820-E` and `KFPO00105-E` messages might be complicated because information about multiple messages cannot be compared. Specifying `Y` in this operand makes it easier to handle such messages because the output messages are classified by error cause. For this reason, it is recommended that you specify `Y` in this operand when you use JP1 to monitor messages.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `N`.

2.2.6 Operands related to full recovery processing

33) pd_max_recover_process = *concurrently-executable-full-recovery-processes-count*

~<unsigned integer>((1-10))<<3>>

Specifies the number of processes to be recovered (`REDO` processes) during full recovery processing. For a HiRDB/Parallel Server, this operand specifies the number of processes to be recovered (`REDO` processes) per server (dictionary server or back-end server).

Condition

You need to use the raw I/O facility.

Specification guidelines

- There are three or more logical volumes for which an RDAREA is defined (per server): 3
- There are fewer than three logical volumes for which an RDAREA is defined (per server): Number of logical volumes
- Increasing the value of this operand increases the input/output concurrency during full recovery processing, and thus can shorten the recovery time. However, because a number of processes equaling *value of this operand* × *server count* are started, determine a value by taking the aforementioned specification value guideline and HiRDB resources into consideration.

34) pd_redo_allpage_put = Y | <u>N</u>

Specifies whether to output to a database the pages that were updated after a synchronization point during full recovery processing.

`Y`:

During full recovery processing, all pages updated after a synchronization point are output.

`N`:

During full recovery processing, only those pages that were not output to the database when an error occurred are output.

Specification guidelines

Normally, there is no need to specify this operand.

When a database is mirrored using LVM (Logical Volume Manager) mirroring facility, the original and duplicate of the mirror might not match each other if an OS or machine error occurs or if a unit error results in system switchover. In this case, to synchronize the original and duplicate volumes:

1. First use LVM facility to synchronize the original and duplicate volumes, and then restart HiRDB.
2. If you cannot use the above method to synchronize the original and duplicate volumes, or if the synchronization operation takes too long to satisfy the system requirements such as the system switchover time, specify `Y` for this operand. During full recovery processing at the restart of HiRDB following an error, all pages that were updated after a synchronization point are output to a database. In this process, HiRDB synchronizes the original and duplicate volumes, eliminating the mismatch between the original and duplicate mirrors.

For the action to take when a mismatch occurs between the original and duplicate mirrors, see the *HiRDB Version 9 System Operation Guide*.

Notes

- When `Y` is specified for this operand, there is overhead for outputting all pages to the database during full recovery processing.
- When this operand is specified, the message `KFPH24004-I` is output when the full recovery processing is finished. No message is output if this operand is omitted.

2.2.7 Operands related to transaction decision processing

35) pd_trn_rerun_branch_auto_decide = Y | N

This operand is applicable only to HiRDB/Parallel Server.

Specifies whether an undecided transaction that has branched from a transaction is to be decided automatically if a unit terminates abnormally before the first prepare processing for transaction commitment control is completed.

`Y`:

Decide undecided transactions automatically.

`N`:

Do not decide undecided transactions automatically. In this case, the HiRDB administrator must make the decision for an undecided transaction. For details about how to decide an undecided transaction, see the *HiRDB Version 9 System Operation Guide.*

Notes

Note the following when `Y` is specified or when specification of this operand is omitted:

- If the reduced activation facility is used, transactions on a unit that is not running at the time of reduced activation are not subject to automatic decision. Therefore, when you restart a unit that was not running during reduced activation, such as after error recovery, check for any undecided transactions. Any undecided transactions that exist must be decided. For details about how to check for undecided transactions and how to decide them, see the *HiRDB Version 9 System Operation Guide*.
- The volume of system log information at the server where the branching-source transaction resides will increase.
- While an undecided transaction is being decided, write-protection is applied to the system log at the server where the branching-source transaction resides.

36) pd_trn_send_decision_intval_sec = *transmission-retry-interval-in-seconds-for-automatic-transaction-decision*
~<unsigned integer>((0-65535)) <<15>> (seconds)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the interval (in seconds) for sending an automatic decision instruction to a branched transaction when the previous send operation has failed, for example, due to a communication error.

Condition

`Y` must be specified in the `pd_trn_rerun_branch_auto_decide` operand, or the `pd_trn_rerun_branch_auto_decide` operand must be omitted.

Notes

Specifying `0` increases the communication workload, because the decision instruction is resent continuously.

Relationships to other operands

This operand has the following relationships with the `pd_trn_send_decision_interval` operand:

- To specify the time before re-transmission in seconds, use the `pd_trn_send_decision_intval_sec` operand; to specify it in minutes, use the `pd_trn_send_decision_interval` operand.
- If the `pd_trn_send_decision_intval_sec` and the `pd_trn_send_decision_interval` operands are both specified, the `pd_trn_send_decision_intval_sec` operand takes precedence.
- If the `pd_trn_send_decision_intval_sec` and the `pd_trn_send_decision_interval` operands are both omitted, the following value is assumed depending on the `pd_sysdef_default_option` operand value:

pd_sysdef_default_option operand value	Default value
`recommendable`	Default value of the `pd_trn_send_decision_intval_sec` operand (15 seconds)
`v6compatible` or `v7compatible`	Default value of the `pd_trn_send_decision_interval` operand (5 minutes)

37) pd_trn_send_decision_interval = *transmission-retry-interval-in-minutes-for-automatic-transaction-decision*

~<unsigned integer>((0-65535)) (minutes)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the interval (in minutes) for sending an automatic decision instruction to a branched transaction when the previous send operation has failed, for example, due to a communication error. Normally, you will specify the `pd_trn_send_decision_intval_sec` operand and omit this operand.

Condition

`Y` must be specified in the `pd_trn_rerun_branch_auto_decide` operand, or the `pd_trn_rerun_branch_auto_decide` operand must be omitted.

Notes

Specifying `0` increases the communication workload, because the decision instruction is resent continuously.

Relationships to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `5`.

This operand has the following relationships with the `pd_trn_send_decision_interval` operand:

- To specify the time before re-transmission in seconds, use the `pd_trn_send_decision_intval_sec` operand; to specify it in minutes, use the `pd_trn_send_decision_interval` operand.
- If the `pd_trn_send_decision_intval_sec` and the `pd_trn_send_decision_interval` operands are both specified, the `pd_trn_send_decision_intval_sec` operand takes precedence.
- If the `pd_trn_send_decision_intval_sec` and the `pd_trn_send_decision_interval` operands are both omitted, the following value is assumed depending on the `pd_sysdef_default_option` operand value:

pd_sysdef_default_option operand value	Default value
`recommendable`	Default value of the `pd_trn_send_decision_intval_sec` operand (15 seconds)
`v6compatible` or `v7compatible`	Default value of the `pd_trn_send_decision_interval` operand (5 minutes)

38) pd_trn_send_decision_retry_time = *maximum-wait-time-for-transaction-auto-decision*

~<unsigned integer>((0-65535)) <<360>> (minutes)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the maximum amount of time to wait for a decision completion response to be returned after an automatic decision instruction has been sent to a branched transaction. If no decision completion response is returned within the amount of time specified here, it is assumed that an unrecoverable error (such as a communication error) has occurred, the decision instruction to the branched transaction is canceled, and the branching-source transaction is decided.

When `0` is specified for this operand, time monitoring is not performed.

Notes

`Y` must be specified in the `pd_trn_rerun_branch_auto_decide` operand, or the `pd_trn_rerun_branch_auto_decide` operand must be omitted.

39) pd_trn_watch_time = *maximum-communication-wait-time-during-transaction-synchronization-point-processing*

~<unsigned integer>((0, 300-65535)) <<3600>> (seconds)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the maximum amount of time to wait for receiving communication (prepare, commit request, or completion communication) between transaction branches during transaction synchronization point processing executed in the HiRDB server process. If no request or completion communication is received within the specified time, the applicable transaction is rolled back if it has not completed the first phase of a two-phase commit. If the first phase has already been completed, the requested processing is performed and the transaction is completed. For details about communications between transactions, see *Commit and rollback* in the manual *HiRDB Version 9 Description*.

Advantage

Even when a HiRDB client halts the transaction determination instruction (by forcibly terminating a client process, for example), the transaction execution continues unless it is stopped by the HiRDB server. Consequently, locked resources in a database, for example, might be monopolized for a long time. Specifying this operand can shorten the time during which locked resources are monopolized.

Specification guidelines

Normally, you need not specify this operand. Specify it in the following cases:

- The `KFPA11989-E` or `KFPA11722-E` message is output during commit.
- Processing of a `COMMIT` statement takes a long time, even though the number of database updates in the transaction is small.

Operand rules

- If `0` is specified, HiRDB waits indefinitely for a prepare and commit request or completion communication.
- If a value between `1` and `299` is specified, it is rounded up to `300`.

Note

- For the commit request or completion communication in the second phase of a two-phase commit, the value specified for this operand takes effect only if `pd_dbsync_point=commit` is specified.

40)pd_trn_rcvmsg_store_buflen = *transaction-recovery-message-queue-size*

~<unsigned integer> ((4096-25600000))<<4096>>(bytes)

When HiRDB performs transaction recovery processing, it registers the transaction to be recovered in the transaction recovery message queue. This operand specifies the transaction recovery message queue size.

Specification guidelines

- Normally, there is no need to specify this operand.
- If the `KFPS00854-W` message (`server=_trnrcv`) is issued during HiRDB operation, consider specifying this operand. The formula for estimating the value to specify is shown below. If the estimated value is 4,096 or less, specify `4096`.
  - ● For HiRDB/Single Server

    $\uparrow 72 \times A \div 1{,}024 \uparrow \times 1{,}024$ (bytes)
  - ● For HiRDB/Parallel Server

$\uparrow 72 \times (B + C + D + E) \div 1{,}024 \uparrow \times 1{,}024$ (bytes)

A: *pd_max_users operand value* × 2 + 7

B:

If the unit contains FES: *b* × 2 + 7

b:

For multi-FES: *pd_max_users operand value* + 1

For non-multi-FES: *pd_max_users operand value*

If the unit contains no FES: 0

C:

If the unit contains a BES (if there are multiple BESs, add the values for individual BESs):

MAX(*pd_max_bes_process operand value*, *pd_max_users operand value*) × 2 + 7

If the unit contains no BES: 0

D:

If the unit contains a DS:

MAX(*pd_max_dic_process operand value*, *pd_max_users operand value*) × 2 + 7

If the unit contains no DS: 0

E:

If the standby-less system switchover (effects distributed) facility is used:

((MAX(*largest pd_max_bes_process operand value in all guest BESs*, *pd_max_users operand value*) × 2 + 7) × *pd_ha_max_act_guest_servers operand value*

If the standby-less system switchover (effects distributed) facility is not used: 0

Notes

- If a value greater than `4096` is specified in this operand, the shared memory size for unit controllers increases. For details, see *Formulas for shared memory used by a unit controller* in the *HiRDB Version 9 Installation and Design Guide*.
- If a value greater than `4096` is specified in this operand, the maximum number of transaction recovery processes (`pdtrnrvd`) increases. The following formula shows how to determine the maximum number of `pdtrnrvd`s.

  ● For HiRDB/Single Server

  MIN(*transaction recovery message queue size* ÷ 72, *pd_max_users operand value* × 2 + 7)

  ● For HiRDB/Parallel Server

  MIN(*transaction recovery message queue size*) ÷ 72, *A* + *B* + *C* + *D*)

  A:

  If the unit contains a FES: *a* × 2 + 7

  a:

  For multi-FES: *pd_max_users operand value* + 1

  For non-multi-FES: *pd_max_users operand value*

  If the unit contains no FES: 0

  B:

  If the unit contains a BES: MAX(*pd_max_bes_process operand value*, *pd_max_users operand value*) × 2 + 7

  If the unit contains no BES: 0

  C:

  If the unit contains a DS: MAX(*pd_max_dic_process operand value*, *pd_max_users operand value*) × 2 + 7

  If the unit contains no DS: 0

  D:

  If the standby-less system switchover (effects distributed) facility is used:

  (MAX(*largest pd_max_bes_process operand value in all guest BESs*, *pd_max_users operand value*) × 2 + 7) × *pd_ha_max_act_guest_servers operand value*

  If the standby-less system switchover (effects distributed) facility is not used: 0

Relationship to other operands

This operand is related to the `pd_max_server_process` operand.

41) pd_trn_commit_optimize = ONEPHASE | NOUSE

This operand is applicable only to HiRDB/Parallel Server.

Specifies whether to use one-phase commit in a HiRDB/Parallel Server's commitment control. For details about one-phase commitment, see the manual *HiRDB Version 9 Description*.

`ONEPHASE`:

Uses *one-phase commit* for commitment control when the number of branches to be updated within a transaction is one (the number of servers to be updated by one transaction is one). Note that using one-phase commit in commitment control is called *one-phase optimization*.

`NOUSE`:

Uses *two-phase commit* for commitment control. One-phase commit is not used.

Specification guideline

Normally specify `ONEPHASE` or omit this operand.

Notes

- Specification of this operand is invalid if an OLTP system has specified two-phase commit.
- The default value of this operand is `NOUSE` if `v6compatible` is specified for the `pd_sysdef_default_option` operand. The default value of this operand is `ONEPHASE` if recommendable or `v7compatible` is specified for the `pd_sysdef_default_option` operand.
- If a recovery-unnecessary front-end server is used, the restrictions on this front-end server takes precedence over the specification for this operand. These restrictions are described below.
  - The log to be output to a front-end server is suppressed.
  - In a recovery-unnecessary front-end server, a UAP that uses the X/Open XA interface to make connection cannot be executed.

42) pd_trn_rollback_msg_interval = *interval-at-which-a-message-indicating-that-rollback-is-underway-is-output*

~<unsigned integer> ((0-1440))<<60>>(minutes)

Specifies the interval at which the `KFPS02235-I` message, indicating that transaction rollback processing is in progress, is to be output. If `0` is specified, the `KFPS02235-I` message is not output.

For details about how to check whether rollback processing is in progress, see the *HiRDB Version 9 System Operation Guide*.

43) pd_trn_rollback_watch_time = *maximum-wait-time-for-rollback-completion-response*[,*rollback-instruction-resending-limit-time*]

This operand is applicable only to HiRDB/Parallel Server.

This operand only takes effect if `Y` is specified in the `pd_trn_rerun_branch_auto_decide` operand or the operand is omitted.

The monitoring target is the UAP for which `0` or nothing is specified for `PDCWAITTIME` in the client environment definition.

***maximum-wait-time-for-rollback-completion-response*:~<unsigned integer>((0-65535))<<3600>> (seconds)**

Specifies the maximum amount of time for the front-end server to wait for receiving a rollback completion response from a back-end server or a dictionary server. If no response is received within the specified time, the rollback instruction is resent to the corresponding back-end server or dictionary server.

Specification guidelines

Normally, there is no need to specify the maximum time to wait for rollback completion response. If your system requires that transactions be decided within 3,600 seconds, specify the maximum time (in seconds) to wait before a transaction is decided, taking into account a situation where transactions cannot be decided.

Notes

- If `0` or nothing is specified for the maximum time to wait for rollback completion response, no rollback instruction is resent.
- If a value from `1` to `9` is specified for the maximum time to wait for rollback completion response, the value is automatically rounded up to 10 seconds.

***rollback-instruction-resending-limit-time*:~<unsigned integer> ((0-65535))<<600>>(seconds)**

Specifies the interval at which the resending of rollback instruction is to repeated if the resending of rollback instruction fails.

If a rollback completion response is not received from the back-end server or the dictionary server within the maximum time to wait for rollback completion response, the front-end server resends the rollback instruction to the corresponding server. If the resending fails, the front-end server keeps resending the rollback instruction until the rollback instruction resending limit time is reached.

If the resending is still not successful when the rollback instruction resending limit time is reached, the `KFPS00936-E` message is issued, the front-end server process terminates abnormally, and then the transaction is canceled.

Specification guidelines

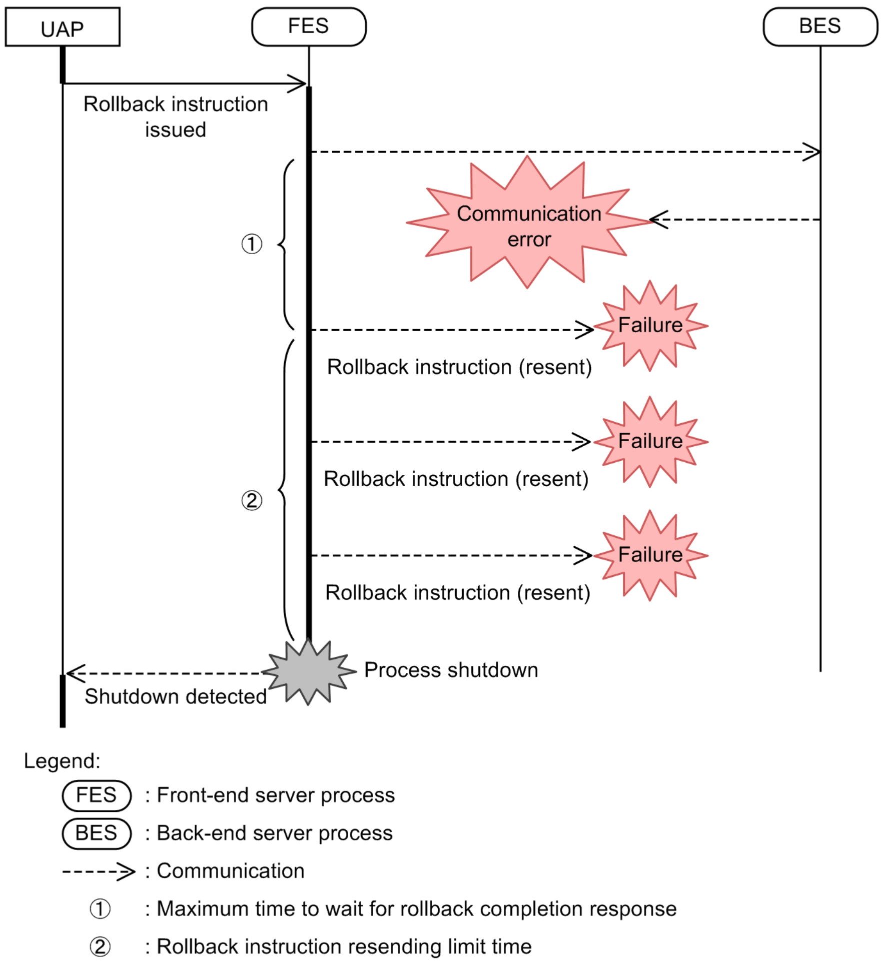
Normally, there is no need to specify the rollback instruction resending limit time. If your system requires that transactions be decided within 600 seconds, specify the maximum time (in seconds) to wait before a transaction is decided, taking into account a situation where the resending of rollback instruction keeps failing.

To send a response to a UAP in the event of a communication failure, *maximum time to wait for rollback completion response* + *rollback instruction resending limit time* is required. Take this into account when specifying the value.

Notes

- If a value from `1` to `239` is specified for the rollback instruction resending limit time, the value is automatically rounded up to 240 seconds.

The following figure shows the scope of this operand.


UAP
FES
BES
Rollback instruction issued
Communication error
①
Failure
Rollback instruction (resent)
Failure
Rollback instruction (resent)
②
Failure
Rollback instruction (resent)
Process shutdown
Shutdown detected
Legend:
FES : Front-end server process
BES : Back-end server process
-----> : Communication
① : Maximum time to wait for rollback completion response
② : Rollback instruction resending limit time

2.2.8 Operands related to the SQL specifications

44) pd_overflow_suppress = Y | N

Specifies whether error suppression is to be implemented during computations. The following types of errors can be suppressed:

- Overflow in the middle of a computation
- Division by 0 errors

`Y`:

If an applicable error occurs in a computation during SQL execution, convert the computation result to the null value and continue the processing.

`N`:

If an applicable error occurs in a computation during SQL execution, cancel the process with an error.

45) pd_space_level = 0 | 1 | 3

Specifies the applicable space conversion level when the space configuration facility is used. For details about the space conversion facility, see the *HiRDB Version 9 System Operation Guide.*

`0`: Use space conversion level 0.

`1`: Use space conversion level 1.

`3`: Use space conversion level 3.

The following table explains the space conversion levels.

Level	Explanation
Level 0	No space conversion.
Level 1	Converts spaces in literals, embedded variables, and ? parameter data in a data manipulation SQL as follows: • If a character string literal is considered a national character string literal, two single-byte spaces are converted into a double-byte space. If a single-byte space appears by itself, it is not converted. • If a character string literal is considered a mixed character string literal, a double-byte space is converted into two single-byte spaces. • When data is being stored into a national character string-type string or is being compared with a national character string-type value expression, two single-byte spaces in an embedded variable or ? parameter are converted into a double-byte space. If a single-byte space appears by itself, it is not converted. • When data is being stored into a mixed character string-type string or is being compared with a mixed character string-type value expression, a double-byte space in an embedded variable or ? parameter is converted into two single-byte spaces.
Level 3	The following processing is performed, in addition to the processing in Level 1: • When data of a national character string-type value expression is being retrieved, a double-byte space is converted into two single-byte spaces.

Notes

- When the space conversion level is changed, there might be differences in UAP results before and after the change. To ensure that UAP results remain the same, do not change the space conversion level.
- When sorting is performed with the space conversion level set to 3, HiRDB performs space conversion on the sorting results, which might produce unexpected results.
- When data is being stored in a cluster key string, space conversion might cause a uniqueness error. In such a case, either store the data without space conversion or use the database reorganization utility to standardize the spaces in the existing database.
- Spaces in a national character string are converted in units of two characters from the beginning of the string.
- Note the following when `1` or `3` is specified for the space conversion level:

  When a storage RDAREA is determined in the case of a UAP that uses a hash function for table partitioning from a hash-partitioned table, it is necessary to specify a space conversion level in the argument of the hash function for table partitioning. If no level is specified, the result of the hash function for table partitioning might be incorrect.

When a UAP is used to execute key range partitioning on a table that has been key range-partitioned (partitioning key is national character data or mixed character data), the partitioning key must be converted with the space conversion function. Otherwise, the result of key range partitioning might be incorrect.

For details about the hash functions for table partitioning and the space conversion function, see the *HiRDB Version 9 UAP Development Guide.*

Relationship to client environment definition

The value of this operand can be different for each client. To change the operand for a client, specify the `PDSPACELVL` operand in the client environment definition.

For details about the `PDSPACELVL` operand, see the *HiRDB Version 9 UAP Development Guide.*

46) pd_dec_sign_normalize = Y | N

Specifies whether the sign of signed packed data transferred from a UAP is to be normalized (in other words, specifies whether the facility for conversion to a `DECIMAL` signed normalized number is to be used).Normalizing the sign means converting the sign `A` to `F` of signed packed data to `C` or `D`. For details about the facility for conversion to a `DECIMAL` signed normalized number, see the *HiRDB Version 9 System Operation Guide.*

`Y`:

Use the facility for conversion to a `DECIMAL` signed normalized number. The sign of signed packed data is normalized. `A` to `F` are considered to be `DECIMAL` data signs.

`N`:

Do not use the facility for conversion to a `DECIMAL` signed normalized number. The sign of signed packed data is not normalized. `C`, `D`, and `F` are considered to be `DECIMAL` data signs.

Notes

The following must be noted when `Y` is specified (that is, when the facility for conversion to a `DECIMAL` signed normalized number is to be used):

- For determining the storage RDAREA for a hash-partitioned table in the case of a UAP that uses the hash function for table partitioning, you must specify to use the facility for conversion to a `DECIMAL` signed normalized number as an argument of the hash function for table partitioning. If this is not specified, the result of the hash function for table partitioning might be corrupted.
- When a UAP is used to execute key range partitioning for a key range-partitioned table (for data whose partitioning key is `DECIMAL`), the partitioning key value must be converted using the facility for conversion to a `DECIMAL` signed normalized number. Otherwise, the result of the key range partitioning might be corrupted.

For details about the hash functions for table partitioning and the facility for conversion to a `DECIMAL` signed normalized number, see the *HiRDB Version 9 UAP Development Guide.*

47) pd_sql_dec_op_maxprec = *maximum-precision-for-a-DECIMAL-type-operation-result-not-exceeding-29-digits*

~<unsigned integer>((29 | 38))

Specifies one of the following maximum precision values, as applicable:

- Maximum precision value when the data type of the operation result is `DECIMAL`
  - Arithmetic operations performed exclusively on data that does not exceed 29 digits
  - Set functions `AVG` and `SUM` performed exclusively on data that does not exceed 29 digits
  - Scalar functions `DECIMAL`, `VALUE`, `GREATEST`, and `LEAST` performed exclusively on data that does not exceed 29 digits
  - `DECIMAL` scalar functions whose argument is `FLOAT` type and that omit precision
  - `CASE` expression when the values of the corresponding `THEN` and `ELSE` clauses do not exceed 29 digits
  - Calls to functions whose only potential functions are user-defined functions that include abstract data type parameters and whose return value data type does not exceed 29 digits
- When the data type of a set operation's result column is `DECIMAL`, the maximum precision value when the column corresponding to the set operation's derived table is exclusively data that does not exceed 29 digits

Specification guidelines

- When installing version 08-04 or later:

  We recommend that you specify `38`.

- In all other cases:
  We recommend that you omit this operand or specifying `29`.

Notes

- If you modify the value of this operand, you must perform the following processing immediately after modification:
  For view tables defined in version 08-04 or later
  Redefine all defined view tables for which any of the `DECIMAL` type operations shown below are specified:
  - Arithmetic operations performed exclusively on data that does not exceed 29 digits
  - Set functions `AVG` and `SUM` performed exclusively on data that does not exceed 29 digits
  - Scalar function `MOD` performed exclusively on data that does not exceed 29 digits
  - `DECIMAL` scalar functions whose argument is `FLOAT` type and that omit precision

  For view tables defined in version 08-03 or earlier
  Redefine all defined view tables.
- Note the following about modifying the value specified in this operand:
  - Changing the operand's value from `29` to `38`

  This change might increase the amount of memory required and the operational overhead. If you are using `DECIMAL` type operations with values never exceeding 29 digits in length, we recommend that you not change this operand value from `29` to `38`.
  - Changing the operand's value from `38` to `29`

  `DECIMAL` type operations on data not exceeding 29 digits in length that worked when `38` was set might overflow when the operand's value is changed to `29`.

Effects on individual estimation formulas

If the value of the `pd_sql_dec_op_maxprec` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Procedure for obtaining the size of the memory required during execution of rapid grouping facility* under *Estimating the memory size required for a HiRDB/Single Server*
- *Procedure for obtaining the size of the memory required during execution of rapid grouping facility* under *Estimating the memory size required for a HiRDB/Parallel Server*

48) pd_sql_mode = 0 | 1

Specifies whether to allow the set operator `MINUS` in SQL statements.

`0`: Do not allow the set operator `MINUS`.

`1`: Allow the set operator `MINUS`.

Notes

- If `1` is specified in this operand, `MINUS` becomes an SQL reserved word. If names such as table and column names contain `MINUS`, change `MINUS` to `"MINUS"` by enclosing it in double quotation marks (`"`). For details about how to specify names, see *Specification of names* in the manual *HiRDB Version 9 SQL Reference*.
- The value of this operand determines whether the set operator `MINUS` is available for use during routine definition and compilation.
- Make sure that the value of the operand specified in the routine definition matches the operand value used when the routine is recompiled. If the value of this operand is changed, and then the routine is recompiled, a syntax error might result.

49) pd_sql_simple_comment_use = Y | N

Specifies whether to allow simple comments in SQL statements.

`Y`: Allow simple comments.

`N`: Do not allow simple comments.

Notes

- If Y is specified in this operand, operations that contain two consecutive minus signs, one representing subtraction and the other representing a unary operator, can no longer be specified without enclosing them in parentheses.
- The value of this operand determines whether simple comments are available for use during routine definition and compilation.
- Make sure that the value of this operand specified in the routine definition matches the value of the operand used when the routine is recompiled. If the value of this operand is changed, and then the routine is recompiled, one of the following problems might occur:
  - A syntax error might occur during recompilation.
  - The execution results obtained from the recompiled routine might differ from those obtained before the routine was recompiled.

2.2.9 Operands related to SQL optimization

50) pd_optimize_level = ***SQL-optimization-option*** **[,*****SQL-optimization-option*****]...**

~<identifier or unsigned integer>

Specifies SQL optimization options. For details about the SQL optimization options, see the *HiRDB Version 9 UAP Development Guide*. The SQL optimization option facilities are explained as follows.

SQL optimization option facility	Identifier	Unsigned integer	S	P
Forced nest-loop-join	"FORCE_NEST_JOIN"	4	Y	Y
Making multiple SQL objects	"SELECT_APSL"	10	Y	Y
Increasing the target floatable servers (back-end servers for fetching data)	"FLTS_INC_DATA_BES"	16	N	Y
Prioritized nest-loop-join	"PRIOR_NEST_JOIN"	32	Y	Y
Increasing the number of floatable server candidates	"FLTS_MAX_NUMBER"	64	N	Y
Priority of OR multiple index use	"PRIOR_OR_INDEXES"	128	Y	Y
Group processing, ORDER BY processing, and DISTINCT set function processing at the local back-end server	"SORT_DATA_BES"	256	N	Y
Suppressing use of AND multiple indexes	"DETER_AND_INDEXES"	512	Y	Y
Rapid grouping facility	"RAPID_GROUPING"	1024	Y	Y
Limiting the target floatable servers (back-end servers for fetching data)	"FLTS_ONLY_DATA_BES"	2048	N	Y
Separating data collecting servers	"FLTS_SEPARATE_COLLECT_SVR"	2064	N	Y
Suppressing index use (forced table scan)	"FORCE_TABLE_SCAN"	4096	Y	Y
Forcing use of multiple indexes	"FORCE_PLURAL_INDEXES"	32768	Y	Y
Suppressing creation of update-SQL work tables	"DETER_WORK_TABLE_FOR_UPDATE"	131072	Y	Y
Deriving search acceleration condition	"DERIVATIVE_COND"	262144	Y	Y
Applying key condition that includes scalar operation	"APPLY_ENHANCED_KEY_COND"	524288	Y	Y
Facility for batch acquisition from functions provided by plug-ins	"PICKUP_MULTIPLE_ROWS_PLUGIN"	1048576	Y	Y

SQL optimization option facility	Identifier	Unsigned integer	S	P
Facility for moving search conditions into a derived table	`"MOVE_UP_DERIVED_COND"`	`2097152`	Y	Y

S: HiRDB/Single Server

P: HiRDB/Parallel Server

Y: Specification is valid.

N: Specification is not applicable.

Operand specification methods

Select the SQL optimization options to be applied and specify their identifiers or unsigned integers. Although either identifiers or unsigned integers (computed values) can be used to specify options, the use of identifiers is usually recommended.

- Using identifiers

  To apply *forced nest-loop-join* and *making multiple SQL objects*, specify the following:

  ```
  pd_optimize_level="FORCE_NEST_JOIN","SELECT_APSL"
  ```

- Using unsigned integers

  To apply *forced nest-loop-join* and *making multiple SQL objects*, specify the following:

  ```
  pd_optimize_level=4,10
  ```

- When you have upgraded from HiRDB Version 5.0 or an earlier version

  The total value specified in HiRDB Version 5.0 or an earlier version is also valid. If there is no need to change the optimization option, you need not change the specification of this operand after upgrading to HiRDB Version 6 or a later version. To add the optimization option, use the following example.

  Example:

  Forced nest-loop-join and *making multiple SQL objects* have been applied to HiRDB Version 5.0, and *priority of OR multiple index use* is to be newly added.

  ```
  pd_optimize_level = 14128
  ```

  However, because this specification makes it difficult to identify the facilities being applied, changing to specification of identifiers is recommended.

Operand rules

- Identifiers and unsigned integers cannot be specified together.
- Identifier specification
  - Enclose each SQL optimization option in quotation marks (").
  - If no SQL optimization option is used, specify `NONE`. When `NONE` and an identifier are both specified, the specification of `NONE` is invalid.
  - Identifiers can be specified in uppercase or lowercase letters.
  - Specifying the same identifier more than once is the same as specifying it once.
- Unsigned integer specification
  - If you do not use the SQL optimization option explained here, specify `0`. However, if both `0` and a non-zero unsigned integer are specified, the specification of `0` is ignored.
  - Specifying the same unsigned integer more than once is the same as specifying it once.

Specification guidelines

For the specification guidelines, see *PDSQLOPTLVL* in the *HiRDB Version 9 UAP Development Guide*.

Operand default value

If this operand is omitted, the following values are assumed:

- HiRDB/Single Server

  ```
  "PRIOR_NEST_JOIN","PRIOR_OR_INDEXES","DETER_AND_INDEXES","RAPID_GROUPIN
  G","DETER_WORK_TABLE_FOR_UPDATE","APPLY_ENHANCED_KEY_COND"
  ```

- HiRDB/Parallel Server

```
"PRIOR_NEST_JOIN","PRIOR_OR_INDEXES","SORT_DATA_BES","DETER_AND_INDEXES
","RAPID_GROUPING","DETER_WORK_TABLE_FOR_UPDATE","APPLY_ENHANCED_KEY_CO
ND"
```

When `v6compatible` is specified in the `pd_sysdef_default_option` operand, `SELECT_APSL` is assumed.

Notes

- If the SQL optimization option is specified in an SQL statement, the SQL optimization specification takes precedence over the specification in this operand. For details about SQL optimization specification, see the manual *HiRDB Version 9 SQL Reference*.
- If the SQL optimization option is specified in an SQL statement (`CREATE PROCEDURE`, `CREATE TYPE`, `ALTER PROCEDURE`, `CREATE TRIGGER`, `ALTER ROUTINE`, or `ALTER TRIGGER`) inside a stored routine or trigger, the SQL optimization option inside the SQL statement takes precedence over the specification in this operand.

Relationship to client environment definition

The value of this operand can be different for each client. To change the value for a client, specify the `PDSQLOPTLVL` operand in the client environment definition. For details about the `PDSQLOPTLVL` operand, see the *HiRDB Version 9 UAP Development Guide.*

Relationship to other operands

- When the `pd_floatable_bes` or `pd_non_floatable_bes` operand is specified in the front-end server definition, specification of *increasing the target floatable servers (back-end servers for fetching data)* and *limiting the target floatable servers (back-end servers for fetching data)* is invalid.
- When `KEY` is specified for the `pd_indexlock_mode` operand, specification of *suppressing creation of update-SQL work tables* is invalid.

51) pd_additional_optimize_level = *SQL-extension-optimizing-option* [,*SQL-extension-optimizing-option*]...
~<identifier or unsigned integer> <<'COST_BASE_2'>>

Specifies SQL extension optimizing options. For details about the SQL extension optimizing options, see the *HiRDB Version 9 UAP Development Guide.*

The following table describes the SQL extension optimizing option facilities.

SQL extension optimizing option facility	Identifier	Unsigned integer
Application of optimizing mode 2 based on cost	`"COST_BASE_2"`	1
Hash execution of a hash join or a subquery#	`"APPLY_HASH_JOIN"`	2
Facility for applying join conditions that include value expressions#	`"APPLY_JOIN_COND_FOR_VALUE_EXP"`	32
Enabling substructure indexes for `XMLEXISTS` predicates that include parameters#	`"ENABLE_INDEX_XMLEXISTS_PARAM"`	256

#: These items are valid when *application of optimizing mode 2 based on cost* is specified.

Operand specification methods

Select the SQL extension optimization options to be applied and specify their identifiers or unsigned integers. Although either identifiers or unsigned integers (computed values) can be used to specify options, the use of identifiers is usually recommended.

- Using identifiers

  To apply *application of optimizing mode 2 based on cost* and *hash execution of a hash join or a subquery*, specify the following:

  ```
  pd_additional_optimize_level= "COST_BASE_2", "APPLY_HASH_JOIN"
  ```

- Using unsigned integers

  To apply *application of optimizing mode 2 based on cost* and *hash-execution of a hash join or a subquery*, specify the following:

  ```
  pd_additional_optimize_level=1,2
  ```

Operand rules

- Identifiers and unsigned integers cannot be specified together.
- Identifier specification
  - ● Enclose each SQL extension optimizing option in quotation marks (").
  - ● Specify `NONE` if the SQL extension optimizing option explained here is not used. However, if both `NONE` and an identifier other than `NONE` are specified, the specification of `NONE` is invalid.
  - ● Identifiers can be specified in uppercase or lowercase letters.
  - ● Specifying the same identifier more than once is the same as specifying it once.
- Unsigned integer specification
  - ● Specify `0` if the SQL extension optimizing option explained here is not used. However, if both `0` and an unsigned non-zero integer are specified, the specification of `0` is invalid.
  - ● Specifying the same unsigned integer more than once is the same as specifying it once.

Specification guidelines

For the specification guidelines, see *PDADDITIONALOPTLVL* in the *HiRDB Version 9 UAP Development Guide*.

Notes

- If SQL optimization is specified in an SQL statement, the SQL optimization specification takes precedence over the specification in this operand. For details about SQL optimization specification, see the manual *HiRDB Version 9 SQL Reference*.
- If the SQL extension optimizing option is specified in an SQL statement (`CREATE PROCEDURE`, `CREATE TYPE`, `CREATE TRIGGER`, `ALTER PROCEDURE`, `ALTER ROUTINE`, or `ALTER TRIGGER`) inside a stored routine or trigger, the SQL extension optimizing option inside the SQL statement takes precedence over the specification in this operand.

Relationship to client environment definition

The value of this operand can be different for each client. To change the value for a client, specify the `PDADDITIONALOPTLVL` operand in the client environment definition. For details about the `PDADDITIONALOPTLVL` operand, see the *HiRDB Version 9 UAP Development Guide.*

Relationship to other operands

When `v6compatible` is specified in the `pd_sysdef_default_option` operand, the default for this operand is `NONE`.

52) pd_hashjoin_hashing_mode = TYPE1 | TYPE2

Specifies the hashing method to be used when `hash join, subquery hash execution` is specified for the SQL extension optimizing option.

`TYPE1`:

This hashing method is used in versions earlier than 07-02.

`TYPE2`:

Provides more uniform hashing than `TYPE1`.

Specification guidelines

- Normally, specify `TYPE2`. However, uniform hashing might not occur depending on the data in the column specified for the join condition. In this case, specify `TYPE1`.
- If specifying `TYPE1` does not produce the expected performance in a user system that has been upgraded to 07-02 or a later version, specify `TYPE2`.

Relationship to client environment definition

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDHJHASHINGMODE` operand in the client environment definition. For details about the `PDHJHASHINGMODE` operand, see the *HiRDB Version 9 UAP Development Guide*.

53) pd_hash_table_size = *hash-table-size*

~<unsigned integer><<256>> (kilobytes)

- 32-bit mode: **((128-524288))**
- 64-bit mode: **((128-2097152))**

Specifies the size of the hash table to be used when *application of hash-execution of a hash join or a subquery* is specified as an SQL optimization option.

Specification guidelines

For details about the hash table size to be specified in this operand, see the *HiRDB Version 9 UAP Development Guide*.

Operand rules

Specify this value as a multiple of 128. If a value that is not a multiple of 128 is specified, the specified value is rounded up automatically to the next multiple of 128.

Relationship to client environment definition

The value of this operand can be different for each client. To change the value for a client, specify the `PDHASHTBLSIZE` operand in the client environment definition. For details about the `PDHASHTBLSIZE` operand, see the *HiRDB Version 9 UAP Development Guide*.

Effects on individual estimation formulas

If the value of the `pd_hash_table_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Procedure for obtaining the size of the memory required during hash join and subquery hash execution* under *Estimating the memory size required for a HiRDB/Single Server*
- *Procedure for obtaining the size of the memory required during hash join and subquery hash execution* under *Estimating the memory size required for a HiRDB/Parallel Server*

54) pd_work_table_option = *work-table-processing-option*

~<unsigned integer> <<1>>

Specifies the HiRDB processing method to be used for executing an SQL statement that uses a work table. Use the following formula to determine the value to be specified for this operand:

Work table processing option = *a* + *b*

Work table processing classification	Variable name	Value	Processing method of HiRDB
Lock acquisition method when AND multiple indexes are used[#1]	*a*	`0`	When the use of AND multiple indexes is selected as the access path, the following occurs: If any of the predicates (data) containing the columns for which the indexes to be used for search are defined satisfies the search condition, that data is locked in the shared mode (PR). Consequently, even if `WITH EXCLUSIVE LOCK` is specified as the lock option[#2], a lock is applied in the shared mode (PR), and not in the exclusive mode (EX).
		`1`	When the use of AND multiple indexes is selected as the access path, the following occurs: If any of the predicates (data) containing the columns for which the indexes to be used for search are defined satisfies the search condition, that data is locked in the specified mode. However, until the search for the first piece of data is completed, a predicate is logically computed for the column for which the index used for the search is defined, and the lock is released from any data that does not satisfy the search condition.
Suppression of message output during automatic extension of the work table buffer[#3]	*b*	`0`	During automatic extension of the work table buffer, the `KFPH29008-I` message is output. This message is output during the first automatic extension of the work table buffer of each server process.
		`8`	During automatic extension of the work table buffer, the `KFPH29008-I` message is not output.

#1

Specifies the HiRDB processing method to be used when using AND multiple indexes.

Use of AND multiple indexes means the following: When a search condition contains multiple conditions connected using AND, and different indexes are defined for each column (for example, `SELECT ROW FROM T1 WHERE C1 = 100 AND C2 = 200`), the multiple indexes are used to create work tables of rows that satisfy the conditions and obtain the product set of these tables.

#2

If no lock option is specified for an SQL statement, WITH SHARE LOCK is normally assumed. However, note that the lock option to be assumed differs in the following cases:

- The FOR UPDATE clause is specified for the cursor.
- The PDISLLVL operand is specified in the client environment definition.
- A data guarantee level is specified for a procedure, facility, or CREATE PROCEDURE, CREATE TRIGGER, CREATE TYPE, ALTER PROCEDURE, ALTER ROUTINE, or ALTER TRIGGER trigger.

#3

As a precondition, the pd_work_buff_expand_limit operand must be specified. For automatic extension of the work table buffer, see the description of this operand.

Specification guidelines

Work table processing classification	Value	Specification guidelines
Lock acquisition method when AND multiple indexes are used	0	When 0 is specified, no lock release is performed, and thus the SQL processing time decreases accordingly. However, if multiple users simultaneously try to update the same table using AND multiple indexes, deadlock might occur. To avoid deadlock, specify 1.
	1	When 1 is specified, lock release is performed, and thus the SQL processing time increases accordingly. However, because the ultimate lock range becomes narrower, the concurrent executability of update-UAPs improves.
Suppression of extension allocation message for work table buffer	0	When 0 is specified, you can monitor whether extension has been allocated for the work table buffer.
	8	By specifying 8, you can reduce the messages output volume.

Notes

Note the following when you specify a lock acquisition mode using AND multiple indexes for work table processing classification:

1. The lock acquisition mode using AND multiple indexes that is applied during stored routine and trigger generation (execution of CREATE PROCEDURE, CREATE TRIGGER, CREATE TYPE, ALTER PROCEDURE, ALTER ROUTINE, or ALTER TRIGGER) is applied to procedures, facilities, and triggers. The lock acquisition mode using AND multiple indexes that is applied during the execution of a stored routine (execution of a CALL statement) or trigger is not applied.
2. If AND multiple indexes are suppressed by the following operands or options, the specification of the pd_work_table_option operand is invalid.
   - pd_optimize_level operand
   - PDSQLOPTLVL operand in the client environment definition
   - SQL optimization option of CREATE PROCEDURE, CREATE TRIGGER, CREATE TYPE, ALTER PROCEDURE, ALTER ROUTINE, or ALTER TRIGGER

Relationship to other operands

When v6compatible is specified in the pd_sysdef_default_option operand, the default for this operand is 0.

55) pd_apply_search_ats_num = *maximum-number-of-combinations-of-narrowing-values-applied-to-ATS-search-condition*

~<unsigned integer>((255-30000))<<255>>

Specifies for an indexed search a maximum number of combinations of narrowing values that can be applied to the ATS or RANGES search condition when an IN predicate or quantified predicate (= ANY (table subquery) or = SOME (table subquery)) is specified.

Specification guidelines

- For initial installation of HiRDB version 08-04 or later:
  We recommend that you specify the value 30000.
- For upgrading of HiRDB version 08-03 or earlier:

We recommend that you specify `255` so that you can use the system under the same conditions as before upgrading.

However, when an `IN` predicate or quantified predicate (= `ANY` (table subquery) or = `SOME` (table subquery)) has been specified in the `RANGE` or `RANGES` SQL statement and the number of combinations of search condition narrowing values exceeds 255, performance might be improved by modifying the value specified for this option. For details about the types of search conditions, see the manual *HiRDB Version 9 Command Reference*.

Note

When the specified value is exceeded, the value is modified so that the search condition becomes different internally, and then the SQL is executed. For this reason, the access path might be other than what was anticipated.

2.2.10 Operands related to narrowed retrieval

56) pd_max_list_users = *number-of-users-who-can-own-lists-concurrently*

~<unsigned integer>((0-32767)) <<0>>

Specifies the maximum number of users who can own lists concurrently.

Relationship to other operands

When this operand is specified, the `pd_max_list_count` operand must also be specified.

Notes

- Do not specify an unnecessarily large value for this operand. Increasing the specification value of this operand increases the size of the shared memory used by HiRDB. For details about the shared memory used by servers, see the *HiRDB Version 9 Installation and Design Guide*.
- If you specify the maximum value (`32767`) for both this operand and the `pd_max_list_count` operand, the size of the shared memory might exceed the upper limit of the size that can be allocated by the OS.

Effects on individual estimation formulas

If the value of the `pd_max_list_users` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula 1* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 1* under *Formulas for the size of the shared memory used by a back-end server*

57) pd_max_list_count = *number-of-lists-created-per-user*

~<unsigned integer>((0-32767)) <<0>>

Specifies the maximum number of lists that one user can create.

Relationship to other operands

When this operand is specified, the `pd_max_list_users` operand must also be specified.

Notes

- Do not specify an unnecessarily large value for this operand. Increasing the specification value of this operand increases the size of the shared memory used by HiRDB. For details about the shared memory used by servers, see the *HiRDB Version 9 Installation and Design Guide*.
- If you specify the maximum value (`32767`) for both this operand and the `pd_max_list_users` operand, the size of the shared memory might exceed the upper limit of the size that can be allocated by the OS.

Effects on individual estimation formulas

If the value of the `pd_max_list_count` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

- *Formula 1* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 1* under *Formulas for the size of the shared memory used by a back-end server*

58) pd_list_initialize_timing = INITIAL | DEFER | STANDBY

Specifies the list initialization (deletion) timing. Normally, lists are initialized when HiRDB is started (including restart). You can use this operand to change the initialization timing. If you create a large number of lists, you might want to consider changing the specification value of this operand.

For details about changing the list initialization timing, see the *HiRDB Version 9 System Operation Guide*.

`INITIAL`:

Lists are initialized during HiRDB startup. HiRDB is started only after all lists have been initialized.

`DEFER`:

Lists are not initialized during HiRDB startup. They are initialized when the `ASSIGN LIST` statement is executed. Consequently, there is some overhead associated with the execution of the `ASSIGN LIST` statement. Consider specifying `DEFER` in the following cases:

- You want to shorten the HiRDB startup time.
- You want to shorten the system switchover time (when user server hot standby or the rapid system switchover facility is being used).

`STANDBY`:

This specification becomes valid when the rapid system switchover facility is used. Even if `STANDBY` is specified when the rapid system switchover facility is not being used, `INITIAL` is used as the default.

Lists are initialized when starting the standby HiRDB. They are not initialized during a system switchover. Lists are not initialized when the `ASSIGN LIST` statement is executed following a system switchover, either. Note that you must create the same lists for both the running system and the standby system.

2.2.11 Operands related to system monitoring

59) pd_utl_exec_time = *utility-execution-monitoring-time*
~<unsigned integer>((0-35791394)) <<0>> (minutes)

Specifies the monitoring time (in minutes) for monitoring the execution time of the following utilities:

- Database load utility (`pdload` command)
- Database reorganization utility (`pdrorg` command)
- Free page release utility (`pdreclaim` command)
- Global buffer residence utility (`pdpgbfon` command)

If the execution of a utility is not terminated within the monitoring time specified by this operand, the executing utility is forcibly terminated, and the error information for identifying the cause of no response is output as shown in the following table.

Error information obtained	Output destination
• Core files • `.deb` files	These files are output in `%PDDIR%\spool\save` of the server machine on which the utility was running.
• `pdls -d rpc -a` command execution result • `pdls -d lck` command execution result	These results are output to the directory specified by the `pd_tmp_directory` operand of the server machine from which the utility command was entered. If the `pd_tmp_directory` operand is omitted, these results are output to the directory specified in the `TMP` system environment variable. If the `TMP` system environment variable is also omitted, they are output in `%PDDIR%\tmp`. However, if HiRDB executes the command, these results are also output in `%PDDIR%\spool\save`. The file names are as follows: *command-nameYYYYMMDDHHMMSSpid*`.txt` • *pid* is a process ID.

Error information obtained	Output destination
	• If the `pdreclaim` or `pdpgbfon` command is executed, HiRDB internally executes the `pdrorg` command, and therefore `pdrorg` is used as the command name to be assigned to the file name.

Advantage

Even if a utility ceases to respond because of an error (communication or disk error, for example) that occurs during utility execution in a nighttime batch job, the succeeding jobs can continue running.

Specification guidelines

- This operand is designed to deal with no-response errors, and not to monitor for large transactions. Therefore, specify a value for this operand that is somewhat greater than the actual maximum execution time of utilities. For example, if the maximum execution time of the database load utility is approximately 60 minutes, and the maximum execution time of the database reorganization utility is approximately 90 minutes, specify `pd_utl_exec_time=120` by leaving some room. In this case, when a process that is normally terminated in 90 minutes does not return a response even after 30 more minutes, it is determined that a no-response error has occurred.
- If this operand is omitted or `0` is specified for it, a utility might go into a no-response state when the following errors occur. Therefore, specify a value other than `0`.
  - Communication error (including temporary error) between servers
  - Process not responding due to a disk error, for example

Operand rules

- If this operand is omitted or `0` is specified for it, utility execution time is not monitored.
- If monitoring time is specified in the `exectime` operand of the `option` control statement of each utility, the specification in the `exectime` operand takes precedence.

60) pd_watch_time = *maximum-wait-time*

~<unsigned integer>((0, 600 to 65535)) <<0>> (seconds)

This operand was provided to ensure compatibility with earlier versions, so it is not necessary to specify it. This operand is applicable only to HiRDB/Parallel Server.

Specifies the maximum amount of time a HiRDB server process is to wait for a response from another HiRDB server process (dictionary server or back-end server). When the set maximum response wait time is exceeded, the HiRDB server process suspends processing (although some HiRDB server processes might not be suspended).

Advantage

If the HiRDB server does not halt execution of an SQL statement, command, or utility, even though the HiRDB client has canceled the SQL statement, command, or utility execution (by forced termination, for example), resources might remain locked for a long time. Specifying this operand can sometimes reduce the period of time this lock is in effect.

Specification guidelines

Specify the largest value among the following time values:

- Time specified by the `PDCWAITTIME` operand of the client environment definition
- Time specified by the `PDLCKWAITTIME` operand of the client environment definition
- Time specified by the `pd_lck_wait_timeout` operand
- Processing time of the SQL statement, command, or utility with the longest execution time

Notes

- If a value in the range `1` to `599` is specified in this operand, the specified value will be rounded up to `600`.
- When the SQL execution time is set as the maximum response wait time, we recommend that you set this operand to `0` (or that the operand be omitted) and that the `PDCWAITTIME` operand of the client environment definition be specified. For details about the `PDCWAITTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.
- If a value smaller than the value set in `PDCWAITTIME` is specified in this operand, additional time might be required for transaction recovery when `pd_watch_time` is exceeded. If this happens, an error will occur, but locked resources will likely not be released in a timely manner.

61) pd_queue_watch_time = ***message-queue-monitoring-time***

~<unsigned integer>((0-3600)) <<600>> (seconds)

This operand is designed to prevent a HiRDB process from not responding. For details about a server process that has stopped responding, see the *HiRDB Version 9 System Operation Guide*.

Reference note

The number of HiRDB server processes is restricted by the following operands:

- `pd_max_server_process`

  If a large number of servers are running within a unit, carefully estimate the value to be specified for this operand. Additionally, if the standby-less system switchover facility is used, the estimated value must also take into account system switchovers.
- `pd_max_bes_process`

  If a multiple front-end server or the standby-less system switchover (1:1) facility is used, carefully estimate the value to be specified for this operand.
- `pd_max_dic_process`

  If a multiple front-end server is used, carefully estimate the value to be specified for this operand.
- `pd_ha_max_server_process`

  If the standby-less system switchover (effects distributed) facility is used, carefully estimate the value to be specified for this operand.
- `pd_max_users`

  If the number of concurrent connections is large, an appropriate value must be specified for this operand.

HiRDB uses a message queue for allocating server processes. When a server process stops responding, messages cannot be extracted from the message queue. If messages cannot be extracted from the message queue within the time specified by this operand (message queue monitoring time), a warning message or error message (`KFPS00888-W` or `KFPS00889-E`) is output. This facility is called the message queue monitoring facility. If one of these messages is output, the server process might have stopped responding.

If `0` is specified in this operand, the message queue is not monitored.

For details about the message queue monitoring facility, see the *HiRDB Version 9 System Operation Guide*.

Operand rule

Because message queue monitoring is carried out in 10-second increments, specify for this operand a multiple of 10 seconds (100 or 110 seconds, for example). If the specified value is not a multiple of 10 seconds, the last digit is rounded up. For example, if you specify 105 seconds, 110 seconds is used.

Relationship to other operands

This operand is related to the `pd_queue_watch_timeover_action` operand.

62) pd_queue_watch_timeover_action = continue | <u>stop</u>

Specifies the processing to be performed by HiRDB when messages cannot be extracted from the message queue within the message queue monitoring time.

`continue`:

HiRDB outputs the warning message (`KFPS00888-W`).

`stop`:

HiRDB outputs the warning and error messages (`KFPS00888-W` and `KFPS00889-E`) and abnormally terminates the unit that has the message queue.

Specification guidelines

- Normally, specify `stop` (or omit this operand). If messages cannot be extracted from the message queue within the message queue monitoring time, the HiRDB server process might have stopped responding. Restarting the unit containing the server process that has stopped responding can sometimes correct the non-responding server process.
- If you do not want to stop HiRDB, specify `continue`. In this case, transactions cannot be executed on the server that has stopped responding. Other servers can still execute transactions. To correct the non-responding server process, use the `pdcancel` command, for example, to terminate the transactions that were being executed in the server process that stopped responding. Afterwards, identify the cause of the server process no-response and take the necessary action. If there were no transactions, use the `pdkill` command to terminate the server that stopped responding. Afterwards, identify the cause of the server

process no-response and take the necessary action. For details about server process no-response and corrective measures, see the *HiRDB Version 9 System Operation Guide*.

63) pd_down_watch_proc = *upper-limit-for-server-process-abnormal-terminations*[,*monitoring-interval*]

This operand is used for monitoring the number of abnormal terminations of a HiRDB server process. Processes to be monitored are those that are abnormally terminated by `PDCWAITTIME` over or aborting.

If abnormal terminations of server processes occur frequently, new services might not be accepted. However, because server process abnormal termination does not cause HiRDB abnormal termination, HiRDB is in an online stopped state, in effect. When this operand is specified, you can pull HiRDB out of this state by restarting it.

***upper-limit-for-server-process-abnormal-termination*: ~<unsigned integer>((0-65535))<<0>>**

If abnormal terminations of server processes exceed the value specified in this operand, HiRDB (applicable unit for a HiRDB/Parallel Server) is abnormally terminated. This is called the facility for monitoring abnormal process terminations. For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

For a HiRDB/Single Server, abnormal terminations of single server processes are counted. In the case of a HiRDB/Parallel Server, the total of the abnormal terminations in the front-end servers, back-end servers, and dictionary servers inside the unit is counted.

If `0` is specified, abnormal terminations of server processes are not monitored.

***monitoring-interval*:~<unsigned integer>((10-3600))<<600>> (seconds)**

Specifies the interval (in seconds) for monitoring abnormal terminations of server processes.

For example, if `100` is specified, abnormal terminations of server processes are monitored every 100 seconds.

Advantages

- Restart of HiRDB refreshes memory and resource statuses, improving the processing efficiency.
- If abnormal termination of server processes occurs frequently, HiRDB is abnormally terminated, and thus the system can be switched over immediately.

Notes

- Do not use this operand in a system that does not allow abnormal termination.
- When a server process is abnormally terminated, the `KFPS01820-E` message is output. Although this message is also output when the server process is abnormally terminated by the `pdcancel` command, this is not counted as an abnormal termination.
- For a mutual system switchover configuration, multiple HiRDBs are activated on the same server machine when system switchover occurs. As a result, the system traffic might increase, causing an adverse effect instead. Therefore, if you specify this operand, we recommend that you restart HiRDB in the system that terminated abnormally.
- If a HiRDB server process terminates abnormally repeatedly, a large amount of troubleshooting information will be output, resulting in frequent input to and output from the HiRDB operating directory, possibly leading to a full disk.

Remarks

- If HiRDB is abnormally terminated by the facility for monitoring abnormal process terminations, the `KFPS01821-E` and `KFPS00729-E` messages are output.
- The following table shows the causes of server process abnormal termination and the server processes that are included in the abnormal termination count.

Cause of server process abnormal termination	Inclusion in abnormal termination count			
	Single server process	Front-end server process	Dictionary server process	Back-end server process
`PDCWAITTIME` operand value of the client environment definition has been exceeded	Y	Y	N[#1]	N[#1]
`pdcancel` command	N	N[#2]	N	N
Internal forced termination (HiRDB internally issues `SIGKILL` and terminates a process)	Y[#3]	Y[#3]	N[#1]	N[#1]
Abort	Y	Y	Y	Y

Cause of server process abnormal termination	Inclusion in abnormal termination count			
	Single server process	Front-end server process	Dictionary server process	Back-end server process
One of the following: • Abnormal termination of server process by transaction recovery processing in an OLTP system • Abnormal termination of server process by XDS transaction recovery processing#4	Y	Y	N	N
Abnormal termination of process other than those described here	Y	Y	Y	Y

Legend:

Y: Included in abnormal termination count

N: Not included in abnormal termination count

#1

If an error is detected in a transaction branch, the abnormal terminations of the front-end server process that has occurred in the same transaction branch are counted.

#2

If the `pdcancel` command is used to forcibly terminate a back-end server process or dictionary server process, the front-end server process is internally and forcibly terminated. In this case, the abnormal termination of the front-end server process might be counted in some cases.

#3

If an error is detected in a global transaction by an OLTP system, the abnormal terminations of the single server process or front-end server process that have occurred in the same global transaction are counted.

#4

If the completion status of a transaction executed from XDS on a server that provides the primary functionality cannot be determined, XDS transaction recovery processing might produce a rollback and the server process might terminate abnormally.

64) pd_host_watch_interval = *host-to-host-monitoring-interval*

~<unsigned integer>((1-180))<<10>> (seconds)

Specifies in seconds the amount of time between monitoring of the operating status of other hosts (server machines).

Specification guidelines

Normally, this operand is omitted.

Notes

- This operand is not applicable to a HiRDB/Single Server or to a HiRDB/Parallel Server that has only one unit.
- If this value is too large, it will be difficult to detect errors at other hosts.
- If this value is too small, a burden will be placed on the network whenever the system has a large number of units, because of the need to generate a high volume of communications for the purpose of monitoring operating status.

Relationship to other operands

This operand is related to the `pd_ipc_conn_nblock_time` operand.

65) pd_watch_resource = <u>MANUAL</u> | AUTO

Specifies whether to output a warning message when the resource usage reaches or exceeds 80%.

`MANUAL`: Do not issue the warning message.

`AUTO`: Issue the warning message.

The following table lists the resources that are monitored.

Monitored resource	Message that is output
Maximum number of concurrent connections specified by the `pd_max_users` operand	`KFPS05123-W`
Concurrently accessible base tables count specified by the `pd_max_access_tables` operand	
Maximum number of RDAREAs specified by the `pd_max_rdarea_no` operand	
Maximum number of HiRDB files comprising an RDAREA specified by the `pd_max_file_no` operand	
Usage of HiRDB file system area for work table files, as specified by the `pdwork` operand	
Maximum number of users who can own lists concurrently, as specified by `pd_max_list_users`	
Maximum number of lists that a user can create, as specified by `pd_max_list_count`	
Number of audit trail files that cannot be specified as swappable targets	
Number of lists created within a server	`KFPH22023-W`

Relationship to other operands

When `AUTO` is specified, 80 is set at the default value for the operands listed below, which eliminates the need to specify these operands (however, if it is necessary to specify a value other than 80 for any of the listed operands, the appropriate value can be specified):

- `pd_max_users_wrn_pnt`#
- `pd_max_access_tables_wrn_pnt`
- `pd_max_rdarea_no_wrn_pnt`
- `pd_max_file_no_wrn_pnt`
- `pdwork_wrn_pnt`
- `pd_max_list_users_wrn_pnt`
- `pd_max_list_count_wrn_pnt`
- `pd_aud_file_wrn_pnt`#
- `pd_rdarea_list_no_wrn_pnt`#

#: 50 is set as the trigger for resetting the warning message output status.

Note

After a version upgrade, the number of resources to be monitored might increase (meaning that other operands in addition to those listed above might be included).Therefore, to keep the monitoring statuses up to date, specify the operands listed above in order to monitor the individual resources, rather than specifying this operand.

66) pd_max_users_wrn_pnt = ***trigger-for-outputting-warning-message-related-to-number-of-connections-to-HiRDB-server*** **[,*trigger-for-resetting-warning-message-output-status*]**

trigger-for-outputting-warning-message-related-to-number-of-connections-to-HiRDB-server **~<unsigned integer>((0-100))<<0 or 80>>(%)**

Specifies when to output the warning message when the number of connections to the HiRDB server reaches or exceeds a predetermined percentage (percentage of the maximum number of concurrent connections specified by the `pd_max_users` operand). For example, if `200` and `90` are specified for the `pd_max_users` and `pd_max_user_wrn_pnt` operands, respectively, the `KFPS05123-W` warning message is output when the number of connections to the HiRDB server reaches or exceeds 180.

Notes

- When `0` is specified, no warning message is issued.
- Specification of this operand is invalid when `9` or less is specified for the `pd_max_users` operand.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the `pd_watch_resource` operand is omitted), `0` is assumed for the `pd_max_users_wrn_pnt` operand (no warning message is issued).
- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, `80` is assumed for the `pd_max_users_wrn_pnt` operand (a warning message is issued when the number of concurrently executing users reaches 80% of the specified maximum).

trigger-for-resetting-warning-message-output-status **~<unsigned integer>((0-99))(%)**

Specifies the trigger for resetting the warning message output status. When the warning message (`KFPS05123-W`) is output, HiRDB goes into the warning message output status. Once HiRDB goes into this status, the warning message is not output again, even if the number of connections to the HiRDB server exceeds the warning message output trigger level again. However, when the number of connections to the HiRDB server falls below the trigger for resetting the warning message output status specified here, the warning message output status is reset.

For example, if `pd_max_users_wrn_pnt=90,70` is specified, the warning message is output when the number of connections to the HiRDB server reaches or exceeds 90% of the maximum number of concurrent connections. Afterwards, no warning message is output until the number of connections to the HiRDB server falls below 70% of the maximum number of concurrent connections. After the percentage falls below 70%, and when it subsequently reaches or exceeds 90% again, the warning message is output.

Notes

- When this specification is omitted, warning-message-output-trigger - `30` is assumed as the default (if the result is a negative number, 0 is used).
- If a value greater than the warning message output trigger is specified, the warning message output trigger value is used.

67) pd_max_access_tables_wrn_pnt = ***trigger-for-issuing-concurrently-accessible-base-tables-count-warning***

~<unsigned integer>((0-100))<<0 or 80>>(%)

Specifies the number of base tables being accessed concurrently that is to trigger issuance of a warning message. The value that is specified is a percentage of the total number of concurrently accessible base tables and sequence generators combined, as specified in the `pd_max_access_tables` operand. For example, if `90` is specified in this operand and `200` in specified in `pd_max_access_tables`, the `KFPS05123-W` warning message will be issued when the total number of base tables and sequence generators being accessed concurrently reaches 180 (90% of 200).

Note

When `0` is specified, no warning message is issued.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the `pd_watch_resource` operand is omitted), `0` is assumed for the `pd_max_access_tables_wrn_pnt` operand (no warning message is issued).
- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, 80 is assumed for the `pd_max_access_tables_wrn_pnt` operand (a warning message is issued when the number of concurrently accessed base tables reaches 80% of the specified maximum).

68) pd_max_rdarea_no_wrn_pnt = ***trigger-for-issuing-RDAREAs-count-warning***

~<unsigned integer>((0-100))<<0 or 80>>(%)

Specifies as a percentage of the maximum number of RDAREAs (as specified by the `pd_max_rdarea_no` operand) the actual number of RDAREAs at which a warning message is to be issued. For example, if `90` is specified for this operand and `200` has been specified for the `pd_max_rdarea_no` operand, a warning message will be issued when the number of RDAREAs reaches 180 (90% of 200). `KFPS05123-W` is issued as the warning message.

Note

When `0` is specified, no warning message is issued.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the pd_watch_resource operand is omitted), `0` is assumed for the `pd_max_rdarea_no_wrn_pnt` operand (no warning message is issued).

- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, `80` is assumed for the `pd_max_rdarea_no_wrn_pnt` operand (a warning message is issued when the number of RDAREAs reaches 80% of the specified maximum).

69) pd_max_file_no_wrn_pnt = *trigger-for-issuing-HiRDB-files-count-warning*

~<unsigned integer>((0-100))<<0 or 80>>(%)

Specifies as a percentage of the maximum number of files comprising an RDAREA (as specified by the `pd_max_file_no` operand) the number of files actually comprising an RDAREA at which a warning message is to be issued. For example, if 90 is specified for this operand and 200 has been specified for the `pd_max_file_no` operand, a warning message will be issued when the number of files comprising an RDAREA reaches 180 (90% of 200). `KFPS05123-W` is issued as the warning message.

Note

When `0` is specified, no warning message is issued.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the `pd_watch_resource` operand is omitted), `0` is assumed for the `pd_max_file_no_wrn_pnt` operand (no warning message is issued).
- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, `80` is assumed for the `pd_max_file_no_wrn_pnt` operand (a warning message is issued when the number of files comprising an RDAREA reaches 80% of the specified maximum).

70) pdwork_wrn_pnt = *trigger-for-issuing-work-table-files-warning*

~<unsigned integer>((0-100))<<0 or 80>>(%)

Specifies as a percentage of the applicable usage# of a HiRDB file system area for work table files (as specified by the `pdwork` operand) the point at which a warning message is to be issued. For example, if `90` is specified for this operand, a warning message will be issued when the applicable usage rate of a HiRDB file system area for work table files reaches 90%. `KFPS05123-W` is issued as the warning message.

#: This operand specifies a percentage of usage in terms of the items listed below, which are specified with the `pdfmkfs` command when the HiRDB file system areas are created; the warning message can be issued for each of these items:

- Usage relative to capacity
- Usage relative to the maximum number of files
- Usage relative to the maximum number of increments

Note

When `0` is specified, no warning message is issued.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the `pd_watch_resource` operand is omitted), `0` is assumed for the `pdwork_wrn_pnt` operand (no warning message is issued).
- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, `80` is assumed for the `pdwork_wrn_pnt` operand (a warning message is issued when the usage reaches 80%).

71) pd_max_list_users_wrn_pnt = *trigger-for-issuing-warning-about-number-of-users-who-have-created-lists*

~<*unsigned* integer>((0-100)) <<0 or 80>> (%)

Specifies as a percentage of the number of users who can create lists (as specified by the `pd_max_list_users` operand) the point at which the number of users actually using lists is to cause a warning message to be issued. For example, if `90` is specified for this operand and `200` was specified for the `pd_max_list_users` operand, a warning message will be issued whenever the number of users using lists reaches 180. `KFPS05123-W` is issued as the warning message.

Notes

- When `0` is specified, no warning message is issued.
- The warning message is output only once. However, if the applicable server is restarted after the warning message has been output, a check of list usage is executed again, and the warning message will be output again.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the `pd_watch_resource` operand is omitted), `0` is assumed for the `pd_max_list_users_wrn_pnt` operand (no warning message is issued).
- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, `80` is assumed for the `pd_max_list_users_wrn_pnt` operand (a warning message is issued when the usage reaches 80%).

72) pd_max_list_count_wrn_pnt = ***trigger-for-issuing-warning-about-number-of-lists-created-by-a-user***

~<unsigned integer>((0-100)) <<0 or 80>> (%)

Specifies as a percentage of the number of lists that can be created per user (as specified by the `pd_max_list_count` operand) the point at which the number of lists created by a user is to cause a warning message to be issued. For example, if `90` is specified for this operand and `200` was specified for the `pd_max_list_count` operand, a warning message will be issued when a user creates 180 lists. `KFPS05123-W` is issued as the warning message.

Notes

- When `0` is specified, no warning message is issued.
- The warning message is output only once. However, if the applicable server is restarted after the warning message has been output, a check of list usage is executed again, and the warning message will be output again.
- If multiple users are creating lists simultaneously, the warning message might be output at the server multiple times, depending on the timing.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the `pd_watch_resource` operand is omitted), `0` is assumed for the `pd_max_list_count_wrn_pnt` operand (no warning message is issued).
- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, `80` is assumed for the `pd_max_list_count_wrn_pnt` operand (a warning message is issued when the usage reaches 80%).

73) pd_rdarea_list_no_wrn_pnt = ***trigger-for-issuing-warning-about-number-of-lists-created-at-server*** **[,*trigger-for-resetting-warning-output-status*]**

trigger-for-issuing-warning-about-number-of-lists-created-at-server **~<unsigned integer>((0-100)) <<0 or 80>> (%)**

Specifies as a percentage of the number of lists that can be created at a server the point at which a warning message is to be issued. For example, if `90` is specified for this operand and the maximum number of lists that can be created at the server is 1,000, a warning message will be issued whenever 900 lists have been created. `KFPH22023-W` is issued as the warning message.

Note

When `0` is specified, no warning message is issued.

Relationship to other operands

- When this operand is omitted and `MANUAL` is specified for the `pd_watch_resource` operand (or the `pd_watch_resource` operand is omitted), `0` is assumed for the `pd_rdarea_list_no_wrn_pnt` operand (no warning message is issued).
- When this operand is omitted and `AUTO` is specified for the `pd_watch_resource` operand, `80` is assumed for the `pd_rdarea_list_no_wrn_pnt` operand (a warning message is issued when the usage reaches 80%).

trigger-for-resetting-warning-output-status **~<unsigned integer>((0-99)) (%)**

Specifies the trigger for resetting the warning message output status.

Whenever the `KFPH22023-W` warning message is issued, HiRDB goes into warning message output status. Once HiRDB is in this status, the warning message will not be issued again even when the number of created lists again reaches the trigger value. However, when the number of created lists falls below the trigger for resetting the warning message output status specified here, the warning message output status is reset.

For example, if `pd_rdarea_list_no_wrn_pnt=90,70` is specified, the warning message will be issued when the number of created lists reaches 90% of the maximum permissible number of lists. Once this occurs, the warning message will not be eligible to be issued again until the number of created lists falls below 70% of the maximum number of permissible lists allowed. Thereafter, the warning message will be issued if the number of lists again reaches the 90% trigger value.

Notes

- If no value is specified for the trigger for resetting the warning output status, `30` is assumed. If a negative is specified, `0` is assumed.
- If the value specified here is greater than the value specified for the trigger for issuing warning, the value specified here is ignored and the same value as was specified for the trigger for issuing warning is assumed as the trigger for resetting the warning output status.

2.2.12 Operands related to SQL runtime warning output facility

74) pd_cwaittime_wrn_pnt = *output-condition-for-SQL-runtime-warning-information (% specification)* | *output-condition-for-SQL-runtime-warning-information (time specification)*

Specify this operand when using the SQL runtime warning output facility. You can specify this operand using one of the following two methods:

- Specifying a percentage
- Specifying a time duration

For details about the SQL runtime warning output facility, see the *HiRDB Version 9 System Operation Guide*.

***output-condition-for-SQL-runtime-warning-information (% specification):* ~<unsigned integer>((0-99)) or <unsigned decimal number>((0-99.999999))<<0>> (%)**

Specifies the condition for outputting SQL runtime warning information of the SQL runtime warning output facility as a percentage of the maximum client wait time (value of the `PDCWAITTIME` operand in the client environment definition). After SQL execution, HiRDB checks the SQL execution time. If this SQL execution time is determined to exceed the time specified by this operand, the following warning information is output. This is called the *SQL runtime warning output facility*.

- SQL runtime warning information file
- Warning message (`KFPA20009-W`)

Operand specification method

Specify this operand as a percentage (%) of the value of the `PDCWAITTIME` operand. For example, if `100` (seconds) is specified for the `PDCWAITTIME` operand and `90` (%) is specified for the `pd_cwaittime_wrn_pnt` operand, HiRDB checks the SQL execution time after SQL execution. If the determined SQL execution time is 90 seconds or longer, but less than 100 seconds, the warning information is output.

Example

```
PDCWAITTIME = 100
pd_cwaittime_wrn_pnt = 90
```

***output-condition-for-SQL-runtime-warning-information (time specification)*: ~<unsigned decimal number>((0-65534.999999))<<0>> (seconds)**

Specifies the condition for outputting SQL runtime warning information of the SQL runtime warning output facility as a time duration. After SQL execution, HiRDB checks the SQL execution time. If this SQL execution time is determined to exceed the time specified by this operand, the following warning information is output. This is called the *SQL runtime warning output facility*.

- SQL runtime warning information file
- Warning message (`KFPA20009-W`)

Operand specification method

Specify the time duration (in seconds) to be used as the output trigger. (You can specify up to the sixth decimal.) Add `sec` to the specified value.

Example

```
pd_cwaittime_wrn_pnt = 0.001sec
```

The following explanation is the same for both percentage and time duration specification.

Operand rule

If `0` or `0sec` is specified in this operand, no warning information is output. (The SQL runtime warning output facility is not used.)

Relationship to client environment definition

You can change the value of this operand for each client. To change it for each client, specify the `PDCWAITTIMEWRNPNT` operand of the client environment definition. For details about the `PDCWAITTIMEWRNPNT` operand, see the *HiRDB Version 9 UAP Development Guide*.

Relationship to other operands

This operand is related to the following operands:

- `pd_cwaittime_report_dir`
- `pd_cwaittime_report_size`

75) pd_cwaittime_report_dir = *SQL-runtime-warning-information-file-output-destination-directory*

~<path name>

Specifies an absolute path name as the output destination directory for the SQL runtime warning information file. Two SQL runtime warning information files (`pdcwwrn1` and `pdcwwrn2`) are created under the directory specified here.

If this operand is omitted, no SQL runtime warning information file is output. However, the warning message (`KFPA20009-W`) is still output.

For a HiRDB/Parallel Server, the SQL runtime warning information files are output to the server machine containing the front-end server to which the UAP that issued the warning target SQL statement is connected.

Operand rules

- Specify no more than 255 characters for the path name.
- The path name is not case sensitive.

76) pd_cwaittime_report_size = *SQL-runtime-warning-information-file-maximum-size*

~<unsigned integer>((2048-2147473627))<<100000>>(bytes)

Specifies the maximum size for the SQL runtime warning information file. The value specified in this operand indicates the size of a single SQL runtime warning information file. Therefore, be careful about the value you specify for this operand because two SQL runtime warning information files are created. For example, if you specify `10,000`, two files, each with a maximum size of 10,000 bytes, are created under the directory.

Specification guidelines

Use the following formula as a guideline when determining the value to be specified for this operand.

{1,280 + *SQL-statement-size* (bytes)} × *number-of-pieces-of-warning-information-to-be-stored-in-file*

If a comment or SQL optimization is specified for the SQL statement, also include the size of the comment and the specified SQL optimization size (bytes) in the SQL statement size.

Remarks

- If the volume of data that is output to an SQL runtime warning information file exceeds the value specified by this operand, the output destination is switched to the other file. The two files are used alternately as this process is repeated. During this process, the oldest information is deleted from the switching destination file.
- If the volume of the SQL runtime warning information that is output at one time exceeds the file size, not all of the SQL runtime warning information is output. Only the information that fits in the file size is output. In this case, a hash mark (#) is added to the end of the SQL runtime warning information.

2.2.13 Operands related to the facility for output of extended SQL error information

77) pd_uap_exerror_log_use = YES | <u>NO</u>

Specifies whether to use the facility for output of extended SQL error information. For details about this facility, see the *HiRDB Version 9 UAP Development Guide*.

`YES`:

Uses the facility for output of extended SQL error information. SQL error information is output in a client error log file and an SQL error report file.

`NO`:

The facility for output of extended SQL error information is not used.

Specification guideline

If you manage SQL error information centrally, or if you output the SQL statements causing an error along with the parameter information, we recommend that you specify `YES`.

Relationship to client environment definition

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDUAPEXERLOGUSE` operand in the client environment definition. If both this operand and the `PDUAPEXERLOGUSE` operand in the client environment definition are specified, the `PDUAPEXERLOGUSE` operand takes precedence.

For details about the `PDUAPEXERLOGUSE` operand, see the *HiRDB Version 9 UAP Development Guide*.

Effects on individual estimation formulas

If the value of the `pd_uap_exerror_log_use` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the size of the memory required to use the facility for output of extended SQL error information* under *Estimating the memory size required for a HiRDB/Single Server*
- *Determining the size of the memory required to use the facility for output of extended SQL error information* under *Estimating the memory size required for a HiRDB/Parallel Server*

78) pd_uap_exerror_log_dir = *SQL-error-report-file-storage-directory*

~<path name of up to 255 characters>

Specifies an absolute path name for the directory in which to store SQL error report files.

Two SQL error report files are created in the specified directory. Their file names are `pduaperrlog1` and `pduaperrlog2`.

If this operand is omitted, no SQL error information is output in an SQL error report file.

For details about the facility to output extended SQL error information, see the *HiRDB Version 9 UAP Development Guide*.

Note

The path name is not case sensitive.

79) pd_uap_exerror_log_size = *SQL-error-report-file-maximum-size*

~<unsigned integer>((2048-2147483647))<<1000000>>(bytes)

Specifies the maximum size of an SQL error report file. The value specified by this operand applies to each of the two SQL error report files that are to be created. When the volume of data that is output to an SQL error report file exceeds the value specified by this operand, the output destination is switched to the other file. The two files are used alternately as this process is repeated. If the volume of the SQL error information that is output at one time exceeds the value specified by this operand, the first through the [*specified value* - 1]-th bytes (up to the 999,999th byte if `1000000` is specified for this operand) of SQL error information is output. In this case, a hash mark (`#`) is added to the end of the SQL error information.

Specification guidelines

Determine the value to be specified for this operand by taking into consideration the volume of SQL error information that you want to retain. You can use the following computation formula:

$(A + B) \times$ *volume-to-be-retained*

- *A* = `1100` + *SQL-statement-size* (bytes)

  This is the size of each piece of SQL error information, excluding the parameter information output size. If an SQL statement contains a comment or the description of SQL optimization specification, the size of the comment or SQL optimization specification also must be included in the SQL statement size. For details about comments and SQL optimization specification, see the manual *HiRDB Version 9 SQL Reference*.

- B = (↑ `pd_uap_exerror_log_param_size`-*operand-value* ÷ 16 ↑ + 1) × 89 × *parameter-count*

  This is the parameter information output size.

80) pd_uap_exerror_log_param_size = *maximum-data-size-of-parameter-information-to-be-output-to-client-error-log-file-and-SQL-error-report-file*

~<unsigned integer>((0-32008))<<0>>(bytes)

Specifies the maximum data size for the parameter information to be output to a client error log file and an SQL error report file.

- When `1` or a value greater than 1 is specified

  Parameter information is output to a client error log file and an SQL error report file.
- When `0` is specified

  Parameter information is not output to a client error log file or an SQL error report file.

If the parameter information is in the variable-length character string type, BLOB type, or BINARY type, the data size area also is included in the specified value.

If the size of the parameter information to be output to a client error log file and an SQL error report file exceeds the value specified for this operand, only the parameter information that fits in the file size is output, and the remainder is discarded.

Relationship to client environment definition

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDUAPEXERLOGPRMSZ` operand in the client environment definition. If both this operand and the `PDUAPEXERLOGPRMSZ` operand are specified, the `PDUAPEXERLOGPRMSZ` operand in the client environment definition takes precedence.

For details about the `PDUAPEXERLOGPRMSZ` operand, see the *HiRDB Version 9 UAP Development Guide*.

2.2.14 Operands related to the SQL reserved word deletion facility

81) pd_delete_reserved_word_file = *SQL-reserved-word-deletion-file-name-1[,SQL-reserved-word-deletion-file-name-2]...*

~<identifier of up to 8 characters>

To use the SQL reserved word deletion facility, specify this operand. For this operand, specify the name of an SQL reserved word deletion file. Create the SQL reserved word deletion file under `%PDDIR%\conf\pdrsvwd`. For details about the SQL reserved word deletion facility, see the *HiRDB Version 9 UAP Development Guide*.

Operand rules

- For the name of the SQL reserved word deletion file, specify any alphanumeric character string (of up to 8 characters) that begins with a letter. The name is not case sensitive.
- You can specify a maximum of 16 SQL reserved word deletion file names.

Format of the SQL reserved word deletion file

***reserved-word*[*,reserved-word*...]**

- When specifying multiple reserved words, use a comma (`,`) as the delimiter. Do not insert a space or tab before or after the comma.
- The maximum length of each definition line is 80 characters. If the definition exceeds 80 characters, use a line feed. However, do not use a line feed in the middle of a reserved word (keyword).
- You can use both lower- and upper-case letters to specify reserved words.
- For details about SQL reserved words, see the manual *HiRDB Version 9 SQL Reference*.

Note

If you delete a reserved word, you can no longer use an SQL function that uses that reserved word. For details about the reserved words that can be deleted using the reserved word deletion facility, see the manual *HiRDB Version 9 SQL Reference*.

Relationship to client environment definition

Specify an SQL reserved word deletion file in the PDDELRSVWDFILE operand in the client environment definition. For details about the PDDELRSVWDFILE operand, see the *HiRDB Version 9 UAP Development Guide*.

2.2.15 Operands related to command execution from SQL

82) pd_sql_command_exec_users = *authorization-identifier*[,*authorization-identifier*]...

~<symbolic name of 8 characters or less>

Specifies the authorization identifiers that can be used to execute commands and utilities with a SQL CALL COMMAND statement. When an authorization identifier is enclosed in double quotation marks, the characters are case sensitive; otherwise, all characters are treated as upper-case.

Note

If an authorization identifier includes a double quotation mark, specify the backslash (\) as an escape character immediately before the double quotation mark.

2.2.16 Operands related to detailed SQLSTATE values

83) pd_standard_sqlstate = Y | <u>N</u>

Specifies whether to output detailed SQLSTATE values.

Y: Outputs detailed SQLSTATE values.

N: Does not output detailed SQLSTATE values.

For details about the SQLSTATE value that is output, see the manual *HiRDB Version 9 Messages*.

2.2.17 Operands related to lock

84) pd_lck_deadlock_info = <u>Y</u> | N

Specifies whether deadlock information, timeout information, and locked resource management table information is to be output. These types of information are output to a directory named %PDDIR%\spool\pdlckinf. For details about deadlock, timeout, and locked resource management table information, see the *HiRDB Version 9 System Operation Guide*.

Y:

Outputs deadlock information, timeout information, and locked resource management table information.

N:

Does not output deadlock information, timeout information, or locked resource management table information.

Relationship to other operands

- When N is specified in the pd_lck_deadlock_check operand, this operand is assumed to be N.
- This operand is related to the pd_lck_deadlock_check operand.

85) pd_lck_wait_timeout = *lock-release-wait-time*

~<unsigned integer>((0-65535))<<Max(180, pd_watch_time value)>>(seconds)

Specifies in seconds the maximum amount of time to wait for lock release (the maximum amount of time in lock release wait status).This is the elapsed time from when a lock release request is placed in wait status until it is released.

If the wait status is not released within the specified amount of time, the SQL statement will return an error. When 0 is specified, lock wait time will not be monitored and waiting will continue until the wait status is released.

Specification guidelines

When you specify this operand, check that the following condition is satisfied:

value specified in the pd_watch_time operand

> *value specified in PDCWAITTIME of client environment definition*

> *value specified in this operand*

Note

If this operand is omitted, the value of the `pd_watch_time` operand might be used as the default value in some cases. Although the `pd_watch_time` operand is invalid, even if it is specified for a HiRDB/Single Server, the specified value might be used as the default value for the `pd_lck_wait_timeout` operand in some cases. Therefore, we recommend that you omit the `pd_watch_time` operand for a HiRDB/Single Server.

Relationship to client environment definition

The value of this operand can be modified for a client. To do so, specify `PDLCKWAITTIME` in the client environment definition. For details about `PDLCKWAITTIME`, see the *HiRDB Version 9 UAP Development Guide*.

Relationship to other operands

- When `v6compatible` is specified in the `pd_sysdef_default_option` operand, the default for this operand is `180`.
- This operand is related to the following operands:
  - `pd_lck_deadlock_check`
  - `pd_lck_deadlock_check_interval`

86) pd_lck_release_detect = interval | pipe

Specifies the method to be used by HiRDB to detect lock release (method of detecting whether the process that locked a resource has released that lock).

`interval`:

Determine the lock release status by checking the lock management area at a regular interval.

`pipe`:

Use a pipe file (process-to-process communication pipe) to receive the lock release notice.

For a HiRDB/Single Server, the default value is `pipe`. For a HiRDB/Parallel Server, the default value is `interval.`

Specification guidelines

Following are guidelines for specifying this operand:

Specified value	HiRDB processing	Advantages and application criteria
`interval`	Determines whether the lock has been released by checking the lock management area in the shared memory. HiRDB checks the lock management area at a regular interval; this interval is specified by the `pd_lck_release_detect_interval` operand.	Even if the process that has locked a resource releases the lock, the release will not be detected until the next time the lock management area is checked. Consequently, a UAP that has a fast processing time per transaction might end up waiting for a long time. However, this wait does not place any load on the CPU or open any file. If a small value is specified for the `pd_lck_release_detect_interval` operand, the CPU usage might increase too much, adversely affecting the throughput. Specifying `interval` has the effect of reducing the CPU load when a slow CPU is being used.
`pipe`	Uses a semaphore to determine whether the lock has been released. The process that has locked a resource sends a lock release notice to the process that is waiting for release. When the process that has locked the resource releases it, the process that is waiting for the release can detect the release	• Throughput improves if the processing time per transaction does not exceed the value specified by the `pd_lck_release_detect_interval` operand. • CPU workload increases if lock-release waits occur frequently. • If a high-speed CPU is used, this method improves throughput.

87) pd_lck_release_detect_interval = *lock-release-detection-interval*

~<unsigned integer>((1-1000))<<10>> (milliseconds)

Specifies the interval at which the lock management area is to be checked.

- When 49 or less is specified

  The interval will begin at the value specified by this operand and thereafter will be 50 milliseconds.
- When 50 or more is specified

  The interval will begin at 50 milliseconds and thereafter will be the value specified by this operand.

Condition

The `interval` option must be specified for the `pd_lck_release_detect` operand.

For a HiRDB/Parallel Server, `interval` is the default value.

Specification guidelines

- As a rule, omit this operand.

  Consider modifying the specified value if transaction processing performance declines significantly because many locks occur in a short period of time and a lock release wait has occurred.
- If too small a value is specified, the CPU workload will increase if lock release waits occur frequently.
- If too large a value is specified, the wait time might increase.
- The value to be specified can be determined by referring to `WAIT TIME` in the statistical information related to system operation from the statistics analysis utility. If the wait time that is output in the statistical information is smaller than the value of this operand, reduce the value of this operand.

88) pd_nowait_scan_option = LOCK | <u>NOLOCK</u>

Specifies the processing method to be used when `WITHOUT LOCK NOWAIT` search is executed.

`LOCK`:

When a `WITHOUT LOCK NOWAIT` search is executed, locking is applied on a row-by-row basis. Only those rows that are not yet being updated or have already been updated by other transactions can be referenced. A row that is being updated by another transaction cannot be referenced until the updating process is finished.

`NOLOCK`:

When a `WITHOUT LOCK NOWAIT` search is executed, locking is not applied. Even those rows being updated by other transactions can be referenced. Note, however, that when a row being updated is referenced, it might not be possible to receive the search target row in some cases.

Note

When you specify `LOCK` for this operand and update a non-fixed table or execute a `WITHOUT LOCK NOWAIT` search, the number of locked resources increases by 1 per transaction. Furthermore, the base row log volume for each piece of data that is output by an update operation using an `UPDATE` statement for a non-fixed table increases by 628 bytes.

Effects on individual estimation formulas

If the value of the `pd_nowait_scan_option` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the amount of base row log information* under *Amount of system log information output during table data updating*

89) pd_lck_queue_limit = *trigger-for-issuing-warning-about-number-of-users-waiting-for-lock-release* ~<unsigned integer>((0-500))<<10>>

Specifies the number of users waiting for lock release of a resource at which time a warning message is to be issued. When the number of users who are waiting for the same resource to be released reaches the specified value, the `KFPS00446-W` warning message is issued. This message is output to the message log file and the event log.

When `0` is specified, no warning message is issued.

90) pd_deadlock_priority_use = Y | <u>N</u>

Specifies whether deadlock priorities are to be used.

`Y`:

Use priority control to determine the program that is to be executed first when deadlock occurs. When deadlock occurs, the SQL of the program with the lower priority is terminated with an error. If both programs have the same priority, the SQL of the program that executed its transaction most recently is terminated with an error. The deadlock priorities are specified with the `PDDLKPRIO` operand of the client environment definition. For details about the PDDLKPRIO operand of the client environment definition, see the *HiRDB Version 9 UAP Development Guide.*

N:

Terminate with an error the SQL of the program that executed its transaction most recently.

91) pd_command_deadlock_priority = 32 | 64 | 96 | 120

Specifies the deadlock priority value of a command.

The value specified for this operand is effective for the following operation commands:

- pdhold -b (reference-possible backup hold)

For details about how to change the deadlock priority value of a command, see the *HiRDB Version 9 System Operation Guide*.

Condition

Y must be specified for the pd_deadlock_priority_use operand.

Specification guideline

If a deadlock occurs in a system that allows a job program to make a retry, you can increase the priority of the operation command to cause an error in the job program. The smaller the specified value, the higher the priority during a lock.

92) pd_key_resource_type = <u>TYPE1</u> | TYPE2

Specifies the method of creating a locked resource for an index key value. For details about how to create a locked resource for an index key value, see the *HiRDB Version 9 UAP Development Guide.*

TYPE1:

Create a locked resource for an index key value by using bit shifts and exclusive-OR operations.

TYPE2:

Create a locked resource for an index key value by using byte-order exclusive-OR operations.

Specification guidelines

- As a rule, TYPE1 is specified. If the key length exceeds 10 bytes, lock conflicts might occur due to synonyms in the locked resource for the index key value. Specification of TYPE1 can prevent this.
- TYPE2 is used for HiRDB Version 5.0, 05-02 or versions earlier than 4.0, 04-05. A user with an upgraded system is recommended to specify TYPE2 if the earlier method does not cause lock conflicts due to synonyms in the locked resource for the index key value.

93) pd_indexlock_mode = {KEY | <u>NONE</u>}

Specifies the locking method for a B-tree index.

KEY: Lock using index key value.

NONE: Do not lock using index key value (executes *index key value no lock*).

For details about index key value no lock, see the *HiRDB Version 9 UAP Development Guide.*

Specification guidelines

Hitachi normally recommends NONE. However, in the event of updating a unique index (adding a deleted key value), either specify KEY or consider operating the system so that there are no residual entries.

Relationship to other operands

This operand is related to the pd_lock_uncommited_delete_data operand. For details, see the explanation of the *pd_lock_uncommited_delete_data* operand.

94) pd_lock_uncommited_delete_data = WAIT | <u>NOWAIT</u>

Specifies the lock release wait control mode for uncommitted delete data.

WAIT:

Use the mode that waits for lock release until the completion status is determined for a transaction that is performing a deletion or update during a search for a line being deleted by a DELETE statement or before an index key being updated by an UPDATE statement is updated, and then returns the search result based on the data after the transaction determination is completed. This also guarantees a uniqueness constraint with the same mode as index key value lock. For details about index key value locks, see the *HiRDB Version 9 UAP Development Guide*.

`NOWAIT`:

Use the mode that returns the search result without waiting until the completion status is determined for a transaction that is performing a deletion or update during a search for a line being deleted by a `DELETE` statement or before an index key being updated by an `UPDATE` statement is updated.

Notes

In the following cases, this operand is ignored even though `WAIT` is specified:

- During searching or updating a dictionary table
- During searching or updating without defining an index in a table for which the `WITHOUT ROLLBACK` option is specified.

When `WAIT` is specified in this operand, the following restrictions apply to the usable size of the HiRDB file that constitutes the table storage RDAREA of the table for which the index is defined:

Page size of table storage RDAREA (bytes)	pd_lock_uncommited_delete_data specification	
	WAIT	NOWAIT
4,096	32 GB	64 GB
6,144	48 GB	
Other	64 GB	

If the HiRDB file already exceeds the usable size or you intend to use a file that exceeds the usable size, modify the page size of the table storage RDAREA to at least 8,192 bytes. In the following cases, continuing operation without modifying the page size will result in an error and the `KFPA19176-E` warning message will be output:

- When allocating an area that will exceed the usable size when a row is added
- When updating or deleting a row stored in a page of an area that exceeds the usable size

Relationship to client environment definition

When `YES` is specified in the `PDLOCKSKIP` operand of the client environment definition, this operand is ignored even if `WAIT` is specified, and search processing using condition evaluation without locking will take priority. For this reason, row deletion does not wait for lock release. However, the indexing for the deleted rows will remains as residual entries.

Relationship to other operands

Some combinations of this operand and the `pd_indexlock_mode` operand cannot be specified, as shown below:

pd_lock_uncommited_delete_data specification	pd_indexlock_mode specification	
	NONE	KEY
`WAIT`	Y	N[#1]
`NOWAIT`	Y	Y
`NOWAIT` (with history of having specified `WAIT`)	Y	N[#2]

Legend:

Y: Combination can be specified.

N: Combination cannot be specified.

#1

If specified, the `KFPS01857-E` error message is output when HiRDB starts.

#2

If HiRDB is started with `WAIT` specified in this operand, subsequently specifying `NOWAIT` in this operand and `KEY` in the `pd_indexlock_mode` operand will result in output of invalid search results when the database is accessed. In this case, execute the database initialization utility (`pdinit`) to initialize HiRDB, and then specify the operand again.

95) pd_dbreuse_remaining_entries = ALL | NONE

If non-locking of index key values is used and index key values are deleted, unneeded entries remain in the index storage pages. In such a case, if `INSERT`, `UPDATE`, or `DELETE` statements are executed in multiple transactions, a lock might occur even when the statement is attempting to manipulate different index key values.

The purpose of this operand is to suppress row identifiers (the values assigned by the system to uniquely identify individual rows) from being reused when row data is inserted, thereby reducing the frequency of locking.

`ALL`:

Reuse row identifiers.

If multiple `INSERT`, `UPDATE`, or `DELETE` statements are executed concurrently and the row identifier of a row to be manipulated or inserted is the same as the row identifier of a remaining entry, locking might occur, even if the target of manipulation is a different index or key value.

`NONE`:

Suppress row identifiers from being reused until the maximum row identifier (255) defined for each page is exceeded.

This can reduce the frequency of locking when different index key values are manipulated by multiple transactions.

Application criterion

If indexes are to be deleted or inserted and a key value insertion processing attempts to reuse a remaining entry that has resulted from index deletion processing, locking might occur. Use this operand if reducing the possibility that locking will occur is more important than reducing the storage efficiency. For details, see *Avoiding locks caused by remaining entries (suppressing row identifiers from being reused)* in the *HiRDB Version 9 UAP Development Guide*.

Specification guidelines

Normally, this operand does not need to be specified.

Consider specifying this operand if all the following conditions are satisfied:

1. Non-locking of index key values is applied (`NONE` is specified in the `pd_indexlock_mode` operand or the operand is omitted).
2. One of the following conditions is satisfied:
   - There is a table for which indexes with `UNIQUE` or `PRIMARY` specified are defined.
   - Locking of uncommitted deleted data (`WAIT` specified in the `pd_lock_uncommited_delete_data` operand) is in effect and the system contains a table with indexes defined.
3. The `INSERT`, `UPDATE`, or `DELETE` statements are executed in multiple transactions on the indexes that satisfy condition 2.

The reusing of row identifiers is suppressed only if the `INSERT` statement is executed on a user table that satisfies the above conditions 1 and 2 and an attempt is made to add rows to the page currently in use. For any other table or operation, row identifiers are reused even if `NONE` is specified.

For details about the effects of this operand on the row identifier allocation method and how a lock-release wait occurs due to remaining entries, see *Avoiding locks caused by remaining entries (suppressing row identifiers from being reused)* in the *HiRDB Version 9 UAP Development Guide*.

Notes

- Entries for indexes remain if unique index keys are updated or deleted while non-locking of index key values is in effect. If uncommitted deleted data are locked and normal index keys are updated and deleted, entries also remain.
- If `NONE` is specified in this operand, the storage efficiency might decreases because the management area inside pages increases. Therefore, if you have changed this operand value, re-estimate the sizes of the RDAREA for table and the RDAREA for indexes. For details about the formulas for estimating these sizes, see *Calculating the number of table storage pages* in the *HiRDB Version 9 Installation and Design Guide*.

  To prevent a decrease in storage efficiency, release free pages periodically by using the free page release utility (`pdreclaim`). For details, see *Reusing used free pages* in the *HiRDB Version 9 System Operation Guide*.

  For details about cases where storage efficiency decreases, see *Avoiding locks caused by remaining entries (suppressing row identifiers from being reused)* in the *HiRDB Version 9 UAP Development Guide*.

- If NONE is specified in this operand, but all 255 row identifiers have been assigned, the row identifiers of the entries remaining in the page are reused sequentially starting with the smallest row identifier. Therefore, if processing that inserts a new index key value attempts to use a remaining entry's row identifier and that remaining entry is still locked (the transaction that performed deletion or update processing has not yet been decided), locking occurs.

  To prevent a decrease in storage efficiency, release free pages periodically by using the free page release utility (pdreclaim). For details, see *Reusing used free pages* in the *HiRDB Version 9 System Operation Guide*.

Relationship to other operands

This operand is related to the following operands:

- pd_indexlock_mode
- pd_lock_uncommited_delete_data

96) pd_lck_deadlock_check = Y | N

Specifies whether checking for deadlock is to be performed.

Y: Checks for deadlocks.

N: Does not check for deadlocks.

Specification guidelines

In transaction systems that do not generate deadlocks, specifying N in this operand can improve SQL execution performance. This is especially true when the interval check mode is used as the deadlock detection method, because lock performance can deteriorate whenever a deadlock is detected if the number of pool partitions for locks is increased. Consequently, we recommend that you specify N in this operand in the case of designing a transaction system that does not generate deadlocks.

In a transaction system that does have potential to generate deadlocks, specify Y in this operand. Specifying N would mean that when a deadlock did occur, SQL would not terminate until the time specified in the pd_lck_timeout operand has elapsed. Also, because HiRDB does not output deadlock information, it might become impossible to determine what caused the deadlock.

Notes

When you specify N in this operand, a transaction that generates a deadlock will not result in an error. Instead, the transaction will be canceled on the basis of one of the following.

- The maximum time for checking lock release wait time elapses and SQL returns an error.
- The maximum wait time of the HiRDB client elapses and the request returns an error to the UAP.

Relationship to client environment definition

This operand is related to the following client environment definitions.

- PDCWAITTIME
- PDLCKWAITTIME

Relationship to other operands

This operand is related to the following operands:

- pd_lck_deadlock_info
- pd_lck_pool_partition
- pd_fes_lck_pool_partition
- pd_lck_deadlock_check_interval
- pd_lck_wait_timeout

97) pd_lck_deadlock_check_interval = *deadlock-check-interval*

~<unsigned integer>((1-2000000000))<<1000>>(msec)

Specifies the interval for performing checking during monitoring for occurrence of deadlocks in the interval check mode.

Condition

Conditions for this operand depend on the server.

For a single server, back-end server, or dictionary server, both the following conditions must be satisfied:

- `Y` is specified in the `pd_lck_deadlock_check` operand
- `2` or greater is specified in the `pd_lck_pool_partition` operand

For a front-end server, both the following conditions must be satisfied:

- `Y` is specified in the `pd_lck_deadlock_check` operand
- `2` or greater is specified in the `pd_fes_lck_pool_partition` operand

Specification guidelines

Specify a small value in this operand to reduce the time between occurrence and detection of a deadlock.

Notes

- If too small a value is specified in this operand, system performance might be degraded.
- If too large a value is specified in this operand, a transaction might be canceled for the following reasons.
  - ● The maximum time for checking the lock release wait time elapses and the SQL statement returns an error.
  - ● The maximum wait time of the HiRDB client elapses and the request returns an error to the UAP.

Relationship to client environment definition

This operand is related to the following client environment definition.

- `PDCWAITTIME`

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_deadlock_check`
- `pd_lck_pool_partition`
- `pd_fes_lck_pool_partition`
- `pd_lck_wait_timeout`

2.2.18 Operands related to buffers

98) pd_sql_object_cache_size = *SQL-object-buffer-size*

~<unsigned integer> <<(pd_max_users value + 3) × 22>>(kilobytes)

- 32-bit mode: **((22-256000))**
- 64-bit mode: **((22-2000000))**

Specifies in kilobytes the size of the buffer area (shared memory) in which SQL objects are to be placed.

Specification guidelines

- SQL objects are saved in a buffer until the user's transaction has terminated. The buffer must be large enough to store the SQL objects of all transactions that will be executed concurrently.
- If the SQL object buffer hit rate is low, the performance might decreases due to the overhead of SQL statement analysis processing.
- SQL analysis can be reduced by saving the SQL objects of static SQLs in the buffer after transaction termination (until the buffer runs out of space) and sharing them among multiple users who execute the same UAP. To effectively utilize the buffer, allocate it so that the SQL objects of frequently used UAPs are resident in the buffer.
- To estimate the buffer length, first determine the buffer length needed for UAP execution from the length of the SQL objects from the SQL statements to be issued by the UAP. Then, compute the buffer length by considering the number of UAPs that will be executed concurrently and the number of concurrently executing users.
- For details about how to estimate the length of the SQL object from a single SQL statement, see *C.2 Formulas for determining size of SQL object buffer (pd_sql_object_cache_size)*.

Tuning the specified value

For details about how to tune the SQL object buffer length, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_sql_object_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula 1* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 1* under *Formulas for the size of the shared memory used by a back-end server*
- *Formula for the shared memory used by a front-end server*

99) pd_def_buf_control_area_assign = <u>INITIAL</u> | TRAN

Specifies whether the control areas for the buffers for table definition information, buffers for view analysis information, buffers for user-defined type information, and buffers for routine definition information are to be allocated in a single batch at the time of HiRDB activation or in a single batch at the time of transaction activation.

`INITIAL`:

Allocate the various buffer control areas in a single batch at the time of HiRDB activation. These control areas are not released until HiRDB terminates.

`TRAN`:

Allocate the various buffer control areas in a single batch at the time of transaction activation. These control areas are released when the transaction is completed.

Specification guidelines

Specify `INITIAL` when performance is important; transaction performance will be improved. However, because a larger shared memory will be required than when `TRAN` is specified, re-evaluate the memory requirements. For details about how to estimate the shared memory size, see the *HiRDB Version 9 Installation and Design Guide.*

Effects on individual estimation formulas

If the value of the `pd_def_buf_control_area_assign` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

100) pd_thread_max_stack_size = *maximum-stack-size-per-thread*

~<unsigned integer>((*0-maximum stack size permitted by the OS* (up to 2047)))<<0>>(megabytes)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the maximum stack size to be used by one thread. When `0` is specified or this operand is omitted, there is no limit to the stack size to be used by one thread. Because HiRDB controls pseudo-threads, it allocates as stack memory that it secures on its own.

Advantages

A UAP error or a complex SQL request from a UAP might result in one thread using a large amount of system resources, causing the OS to hang up. When this operand is specified, it limits the stack size to be used by one thread, thus preventing the OS from hanging up.

Notes

The value specified by this operand must be equal to or smaller than the maximum stack size permitted by the OS.

101) pd_thread_stack_expand_size = *stack-extension-size-per-thread*

~<unsigned integer> ((0-2146435072))<<0>>(bytes)

This operand increases the stack size per thread for the threads controlled by HiRDB.

Normally, this operand is omitted. Change the value of the operand only if asked to by a maintenance engineer.

2.2.19 Operands related to shared memory

102) pd_shmpool_attribute = free | fixed

Specifies whether to fix the shared memory used by the HiRDB unit controller in memory. Note that whether this feature is supported depends on the version of Windows you are using. For details about the prerequisites for Windows, see *page locking shared memory* under *System design* in the *HiRDB Version 9 Installation and Design Guide*. If this feature is not supported, this operand is ignored even if `fixed` is specified.

`free`:

Do not fix the shared memory in memory. Depending on the size of the real memory, another page of shared memory might be created, affecting performance adversely.

`fixed`:

Fix the shared memory in memory.

Advantage

Fixing the shared memory to be used by the HiRDB in the memory (`fixed` specified) prevents shared memory paging, thus improving the access performance to the shared memory.

Specification guidelines

Determine whether to fix shared memory based on the computed shared memory size and the real memory size of the server machine. Because there is a limit to the amount of memory that can be fixed for the HiRDB, fixing too much shared memory might cause frequent paging of other types of memory. How much real memory there is and the amount of memory to be occupied by the shared memory pool need to be carefully evaluated.

For the formulas used to calculate the shared memory sizes to be used by the unit controller and individual servers, see the *HiRDB Version 9 Installation and Design Guide*.

Notes

The following notes apply when `fixed` is specified in a Windows operating system that supports page fixing:

- The shared memory must be allocated to the paging file with the `pdntenv` command. For details about the `pdntenv` command, see the manual *HiRDB Version 9 Command Reference*.
- When you estimate the required memory size, note that the shared memory is acquired with its size rounded up to the page size of a Windows large page. To check the page size of a Windows large page, use the `pdntenv -os` command.

103) pd_shmpool_control = unit | server

This operand is applicable only to HiRDB/Parallel Server.

Specifies the control method for shared memory used by HiRDB.

`unit`:

Applies the unit-by-unit control method for starting and stopping the HiRDB/Parallel Server.

`server`:

Applies the user-server-by-user-server control method for starting and stopping the HiRDB/Parallel Server (when using the `pdstart` or `pdstop` command with the `-s` option specified).

User server refers to a front-end server, dictionary server, or back-end server.

Specification guidelines

- Normally, `unit` is specified in this operand or this operand is omitted.
- When performing a user server start or stop operation on a system that does not use standby-less system switchover (effects distributed), refer to *Securing and releasing shared memory* below, and then consider whether to specify `server` in this operand.

Securing and releasing shared memory

Shared memory used by HiRDB is secured as a block from the OS when the unit starts and is then managed by the HiRDB in a shared memory pool. When a user server starts, the portion of shared memory used by that user server is secured from the shared memory pool in multiple parts.

When the user server shuts down, the shared memory secured when that server started is released.

Accordingly, when the start and stop operations of a HiRDB/Parallel Server are performed for individual user servers, fragmentation can arise in the area within the shared memory pool. As fragmentation increases, units might terminate abnormally because they cannot secure shared memory when the user server starts. If `server` is specified in this operand, the shared memory required for a user server to start can be secured despite fragmentation.

If a unit is using the standby-less system switchover (effects distributed) facility, the HiRDB is controlled so that fragmentation in the shared memory pool area does not occur, even though user-server start and stop operations are performed. Consequently, there is no need to specify `server` in this operand when performing start and stop operations only for back-end servers within units that use the standby-less system switchover (effects distributed) facility.

104) pd_dbbuff_attribute = <u>free</u> | fixed

Specifies whether to fix the shared memory used by the global buffer in the real memory. Note that whether this feature is supported depends on the version of Windows you are using. For details about the prerequisites for Windows, see *page locking shared memory* under *System design* in the *HiRDB Version 9 Installation and Design Guide*. If this feature is not supported, this operand is ignored even if `fixed` is specified.

`free`:

Do not fix the shared memory to be used by the global buffer in the real memory.

`fixed`:

Fix the shared memory to be used by the global buffer in the real memory.

Advantage

Fixing the shared memory to be used by the global buffer in the real memory ("fixed" specified) prevents shared memory paging, thus improving the performance of accesses to the shared memory.

Specification guidelines

- To emphasize performance when there is ample real memory, specify `fixed`.
- Determine whether to fix shared memory based on the computed shared memory size and the real memory size of the server machine. If a memory page that is quite large relative to the real memory is fixed, the result might be frequent paging or a virtual memory shortage. Therefore, how much real memory there is and the amount of memory to be occupied by the shared memory pool need to be evaluated carefully. For the formulas used to calculate the shared memory size to be used by the global buffer, see the *HiRDB Version 9 Installation and Design Guide*.
- When the size of the shared memory to be page-fixed is subtracted from the size of the real memory, as a guideline, make sure that the result does not equal or exceed half the size that is obtained by subtracting the size of the shared memory to be page-fixed from the swap area size.

Relationship to other facilities

- If `fixed` is specified, the shared memory used by a dynamically modified global buffer is also fixed in the real memory. Therefore, carefully consider the real memory size before adding or modifying a global buffer.

Notes

The following notes apply when `fixed` is specified in a Windows operating system that supports page fixing:

- The shared memory must be allocated to the paging file with the `pdntenv` command. For details about the `pdntenv` command, see the manual *HiRDB Version 9 Command Reference*.
- When you estimate the required memory size, note that the shared memory is acquired with its size rounded up to the page size of a Windows large page. To check the page size of a Windows large page, use the `pdntenv -os` command.
- In the `SHMMAX` operand, specify a value that is rounded up to the page size of a Windows large page.
- If the shared memory for the global buffer pool cannot be fixed in memory when HiRDB starts, the `KFPH23045-W` message is issued. In this case, even if `fixed` is specified, HiRDB resumes processing without page-fixing some, or all, of the shared memory in memory.

105) pd_shared_memory_report = <u>Y</u> | N

This operand is applicable only to HiRDB running in 32-bit mode.

This operand specifies whether shared memory-related report messages are to be issued when HiRDB starts.

`Y`: Output report messages.

`N`: Do not output report messages.

The following table lists and describes the shared memory subject to reporting and the messages that are issued:

No.	Target shared memory	Message that is issued	Overview of message
1	Shared memory for global buffers	`KFPH23048-I` message	This message reports the size of the shared memory for global buffers that has been allocated.
2		`KFPH23049-W` message	This message is issued when the shared memory for global buffers that has been allocated exceeds 1.2 gigabytes.
3		`KFPH23050-I` message	This message reports the maximum size of the management area that is used by each shared memory for global buffers on the server.
4		`KFPH23051-W` message	This message is issued if the specified `SHMMAX` operand value exceeds 200 megabytes (50 megabytes if the facility for dynamically changing global buffers (`pd_dbbuff_modify=Y`) is used).
5	Shared memory for unit controller	`KFPS05604-I` message	This message reports the size of the shared memory for unit controllers that has been allocated.

Specification guidelines

Normally, this operand does not need to be specified.

If `N` is specified in this operand, none of the above five messages are issued. If you only want to suppress the output of warning messages (for reasons such as the warning messages might affect the output of event monitoring messages), consider suppressing the output of messages to the event log (by using the `pdmlgput` operand). For details about the suppression of message output to the event log, see *Suppressing message output to the event log* in the *HiRDB Version 9 System Operation Guide*.

Notes

- This function is also enabled when HiRDB is run in 32-bit emulation mode in 64-bit mode Windows. Warning messages might be issued, but ignore them. For details about how to suppress the output of these messages, see *Specification guidelines*.
- The `KFPH23049-W` message is issued if the size of shared memory for global buffers exceeds 1.2 gigabytes. If the `KFPH23049-W` message is issued, HiRDB start processing might fail because the shared memory cannot be attached by HiRDB's process, the unit might be shut down during online operation due to insufficient memory, and it might not be possible to restart HiRDB. See the actions to be taken for the `KFPH23049-W` message to determine how to respond to the message.
- The `KFPH23051-W` message is issued if a value greater than 200 megabytes (if the facility for dynamically changing global buffers (`pd_dbbuff_modify=Y`) is used, 50 megabytes) is specified for the maximum size of shared memory segment (value of the `SHMMAX` operand). If the `KFPH23051-W` message is issued, HiRDB start processing might fail because the shared memory cannot be attached by HiRDB's process, the unit might be shut down during online operation due to insufficient memory, and it might not be possible to restart HiRDB. See the actions to be taken for the `KFPH23051-W` message to determine how to respond to the message.

106) pd_max_dbbuff_shm_no= ***maximum-number-of-shared -memory-segments-for-global-buffer***

~<unsigned integer> ((16-512))<<16>>

This operand is applicable only to HiRDB running in 32-bit mode.

If the shared memory for global buffers is allocated based on the segment size specified in the `SHMMAX` operand when HiRDB starts, this operand specifies the maximum number of segments to be allocated per server.

Specification guidelines

If facility for dynamically changing global buffers is used, the values of this operand and the `SHMMAX` operand cannot be changed when HiRDB is restarted. If these operand values are too large, the restart processing might fail because the required area cannot be allocated. To avoid this, specify the value obtained from the formula shown below in this operand. For the `SHMMAX` operand, specify a value of `50` or smaller.

↑ *shared memory required for global buffer (megabytes)* ÷ *SHMMAX operand value* ↑

Notes

- If the facility for dynamically changing global buffers is not used, you can increase the value of this operand when HiRDB is restarted. If the value of this operand is reduced, the restart processing will fail.
- If the facility for dynamically changing global buffers is used, the total number of shared memory segments that are allocated for global buffer can be obtained from the following formula:

  pd_max_dbbuff_shm_no value + pd_max_add_dbbuff_shm_no value

  When HiRDB starts, if more shared memory segments are needed than the number specified for `pd_max_dbbuff_shm_no`, the required shared memory segments are allocated by the facility for dynamically changing global buffers. Therefore, the number of additional shared memory segments that can be allocated by the facility for dynamically changing global buffers is less than the number specified for `pd_max_add_dbbuff_shm_no`. This value can be determined from the following formula:

  Value of pd_max_add_dbbuff_shm_no - number of shared memory segments that were allocated beyond the value specified for pd_max_dbbuff_shm_no
- If you have changed the value of the `pd_max_dbbuff_shm_no` operand, re-estimate the number of resources (number of shared memory segments) required for the `PDUXPLSHMMAX` system environment variable.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pd_dbbuff_modify`
- `pd_max_add_dbbuff_shm_no`
- `PDUXPLSHMMAX` system environment variable

Effects on individual estimation formulas

If the value of the `pd_max_dbbuff_shm_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*

2.2.20 Operands related to message log files

107) pd_mlg_msg_log_unit = <u>manager</u> | local

This operand is applicable only to HiRDB/Parallel Server.

Specifies a message log output destination unit.

`manager`:

Outputs message logs in the system manager unit's message log file and the event log. The message log file is created under `%PDDIR%\spool` of the system manager's server machine.

`local`:

Outputs message logs in the message log file and the event log of the unit that sent the message. The message log file is created under `%PDDIR%\spool` of each server machine.

Specification guidelines

- To use the system manager unit to centrally manage message logs, specify `manager` for this operand.
- If an error or other factor has caused the system manager to stop, message logs might be output after a time delay in the event log of the unit that sent the message, or messages themselves might not be output at all. If you specify `local` for this operand, messages can be output to the unit that sent the message without any delay. For details about how to distribute message log output, see the *HiRDB Version 9 System Operation Guide*.

The following table shows the relationship between the value of this operand and the message log output destination.

pd_mlg_msg_log_unit value	Condition	Message log output destination
`manager`	Normal operation	Message logs are output in the message log file and the event log of the server machine where the system manager is located.
	System manager unit error or communication error	Message logs are output in each server machine's the event log. Some message logs might not be output. Message logs might be output with a delay.
`local`	Normal operation	Message logs are output in the message log file and the event log of each server machine.
	System manager unit error or communication error	

Relationship to other operands

If you specify `local` for this operand, estimate the size of the message log file again and specify the result for the `pd_mlg_file_size` operand.

Effects on individual estimation formulas

If the value of the `pd_mlg_msg_log_unit` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*

108) pd_mlg_file_size = *maximum-message-log-file-size*

~<unsigned integer>((64-1000000))<<1024>>(kilobytes)

Specifies in kilobytes the maximum size of a message log file.

Two message log files are provided. When the amount of stored message information reaches the size specified here, the message log files are swapped.

If you specify `local` for the `pd_mlg_msg_log_unit` operand, a message log file having the size specified here is created in each server machine.

2.2.21 Operands related to statistical information

109) pd_statistics = Y | <u>N</u>

Specifies whether collection of the statistics log is to be started at the time of HiRDB startup.

`Y`:

Begin collecting the statistics log at the time of HiRDB startup.

`N`:

Do not begin collecting the statistics log at the time of HiRDB startup.

When `Y` is specified, statistics log information on the system operation of the overall unit only is collected. If information on each server or information other than the statistics information on system operation is needed, the `pdstend` command must be entered to stop statistical information collection after HiRDB startup has been completed, then the `pdstbegin` command can be entered to begin collecting statistical information again.

When `N` is specified, it is still possible to collect a statistics log during HiRDB operation by entering the `pdstbegin` command.

Notes

The statistics log information cannot be acquired in the following cases:

- The `pdstbegin` operand is specified.

- The standby-less system switchover (1:1) facility is used and the system is switched to an alternate BES unit.
- A unit to which the standby-less system switchover (effects distributed) facility is applied
- No server has started in the unit.

110) pd_stj_file_size = *maximum-statistics-log-file-size*

~<unsigned integer>((64-1000000))<<1024>>(kilobytes)

Specifies in kilobytes the maximum size of a statistics log file.

Two statistics log files are provided. When the amount of stored statistics log information reaches the size specified here, the statistics log files are swapped.

Specification guidelines

- For details about how to determine the value to be specified for this operand, see *C.1 Formulas for determining size of statistics log file (pd_stj_file_size)*.
- Specify this operand such that the following relationship with the `pd_stj_buff_size` operand is maintained:
  `pd_stj_file_size` ≥ `pd_stj_buff_size` × `2`

111) pd_stj_buff_size = *statistics-log-buffer-size*

~<unsigned integer>((32-512)) <<32>> (kilobytes)

Specifies the statistics log buffer size.

Specification guidelines

The default value of 32 is appropriate when the following types of statistical information are not to be output:

- SQL object execution information
- Statistics about SQL object transmission

If these types of statistical information are to be output, specify the value obtained by adding 32 to the result obtained from the following formula (however, if the result exceeds 512, specify 512):

$(a \div 1{,}024) \times (0.03 \div b)$

a: Statistics log output volume (bytes)

For details, see *C.1 Formulas for determining size of statistics log file (pd_stj_file_size)*.

b: Statistical information output time (seconds)

112) pd_sqlobject_stat_timing = <u>deallocate</u> | tran

Specifies the timing for collecting statistics on an SQL object buffer. The following table lists and describes the timing for collecting the statistical information and the statistical information that is collected:

No.	Timing for collecting statistical information	Statistical information that is collected	
		deallocate	tran
1	When the preprocessing results become invalid if the following SQL statements are executed: • `DEALLOCATE PREPARE` statement • `PREPARE` statement • `EXECUTE IMMEDIATE` statement	Preprocessing results that are invalidated	Preprocessing results that are invalidated
2	When committed	Preprocessing results that are invalidated by commit	All preprocessing results
3	When rolled back	Not collected	All preprocessing results
4	When the `DISCONNECT` statement, `COMMIT` statement with `RELEASE` specified, or a definition SQL statement is executed	All preprocessing results	All preprocessing results
5	When the `ROLLBACK` statement with `RELEASE` specified is executed	Preprocessing results that are not invalidated by rollback when the `ROLLBACK` statement with `RELEASE` specified is executed	All preprocessing results

Specification guidelines

Specify `tran` in the following cases:

- The preprocessing results become valid beyond the scope of a transaction such as when `TRUE` is specified for `STATEMENT_COMMIT_BEHAVIOR` in a holdable cursor or the JDBC driver.
- The statistics on an SQL object buffer is to be collected during rollback processing.

In any other case, specify `deallocate`.

Notes

- When a transaction is decided due to the cancellation of a server process, the statistical information cannot be collected.
- If `TRUE` is specified for `STATEMENT_COMMIT_BEHAVIOR` in a holdable cursor or the JDBC driver, the preprocessing results remain valid beyond the scope of the transaction. Therefore, if `deallocate` is specified in this operand, the statistics on these SQL object buffers cannot be collected even when the transaction is decided.
- If `TRUE` is specified for `STATEMENT_COMMIT_BEHAVIOR` in a holdable cursor or the JDBC driver, the statistics on an SQL object buffer can be collected only while that cursor is closed.

For details about the holdable cursor and `STATEMENT_COMMIT_BEHAVIOR`, see the *HiRDB Version 9 UAP Development Guide*.

2.2.22 Operands related to RPC trace information

113) pd_rpc_trace = Y | N

Specifies whether RPC trace information is to be collected. HiRDB maintenance information is output in the RPC trace information.

As a rule, omit this operand.

`Y`: Collect RPC trace information.

`N`: Do not collect RPC trace information.

Note

Specifying `Y` for this operand degrades communication performance.

114) pd_rpc_trace_name = *"name-for-RPC-trace-collection-files"*

~<path name of up to 254 characters> <<%PDDIR%\spool\rpctr>>

Specifies as an absolute path name the file name for the RPC trace files. Three RPC trace files are created, with `1`, `2`, and `l` suffixed to the specified file name.

Note

- Files with a maximum size of `pd_rpc_trace_size value × 2` are created under the directory specified by this operand. Pay attention to the file size.

115) pd_rpc_trace_size = *RPC-trace-collection-file-size*

~<unsigned integer>((1024-2147483648))<<4096>>(bytes)

Specifies the size of the RPC trace files in bytes.

The value specified by this operand determines the size of both RPC trace file 1 and RPC trace file 2.

Specification guidelines

An RPC trace collects in a single file in time order the information on the communications among all processes. If the volume of this information exceeds the specified value, trace data output is switched to the other file, which means that older trace information will be overwritten. When this happens, there might not be enough trace information available, making troubleshooting difficult. For this reason, specify at least `1000000` for this operand.

Note

The value specified by this operand does not apply to RPC trace file *l*, because *l*'s size is fixed at 0 bytes.

2.2.23 Operands related to performance trace information

116) pd_prf_trace = <u>Y</u> | N

Specifies whether performance trace information is to be collected. For details about the performance trace facility, see *Performance Trace Facility* in the *HiRDB Version 9 System Operation Guide*.

`Y`:

Collect performance trace information.

`N`:

Do not collect performance trace information. If performance trace information is not collected, the `pdprfd` process is not started.

Specification guidelines

Normally, this operand does not need to be specified. We recommend that you do not specify `N` because, in the event of an error, this setting makes it difficult to determine the cause of the error.

Notes

If `N` is specified in this operand, the setting cannot be changed to collect performance trace information while HiRDB is running. To collect performance traces, terminate HiRDB, enable the collection of performance traces, and then restart HiRDB.

117) pd_prf_level = <u>00000007</u> | 0000001f | 0000007f | 000001ff | 00000000

Specifies the performance trace information collection level in hexadecimal notation. For details about the information that can be collected according to the performance trace collection level, see *Details about performance trace information and collection points* in the *HiRDB Version 9 System Operation Guide*.

The following table describes the collection level determined by the value of the `pd_prf_level` operand and the information that is collected.

Table 2–1: Collection level determined by the value of the pd_prf_level operand and the information that is collected

Value of the pd_prf_level operand	Collection level	Positioning	Information that is collected
`00000007` (Minimum level)	`00000004`	Provides the minimum amount of information	Traces of `CONNECT` operation, utility execution, and specific events
`0000001f` (Default level)	`00000004`, `00000010`	Allows the boundaries with HiRDB (entry and exit) to be identified	Traces mostly before and after communication with HiRDB in addition to the traces obtained at the minimum level
`0000007f` (Detail level)	`00000004`, `00000010`, `00000040`	Provides the information required for tracing SQL processing	Traces that can be used to check the flow of HiRDB's internal processing in addition to the traces obtained at the default level
`000001ff` (Maintenance level)	`00000004`, `00000010`, `00000040`, `00000100`	Provides the maintenance information that is useful in the event of an error	Traces that includes iteration processing such as internal locking in addition to the traces obtained at the detail level
`00000000` (Suppression level)	Not collected	Suppresses the collection of trace data	Not collected

Specification guidelines

- We recommend that you specify `0000001f` (default level) so that when you investigate performance-related errors, you can quickly identify the cause of the problem. If a decrease in online performance is acceptable, such as during environment configuration, consider specifying `0000007f` (detail level).
- Normally there is no need to specify `000001ff` (maintenance level). Specify this level only if you are requested to do so by the support service.

- We recommend that you do not specify `00000000` (suppression level) because, in the event of an error, this setting makes it difficult to determine its cause.

Notes

- If the number of performance trace collection points is increased or the collection level is changed to acquire more detailed performance traces, online performance might decrease.
- Do not specify a value other than `00000007`, `0000001f`, `0000007f`, `000001ff`, or `00000000`. If any other value is specified, it becomes more difficult to verify performance and investigate errors.

Relationship to other operands

This operand is related to the following operands:

- `pd_prf_file_count`
- `pd_prf_file_size`

Remarks

By using the `pdprflevel` command, you can change the collection level even while HiRDB is running.

118) pd_prf_file_count = *number-of-performance-trace-information-file-generations*

~<unsigned integer> ((3-256))<<15>>

Specifies the number of performance trace information file generations that will be retained.

Specification guidelines

- Normally, this operand does not need to be specified.
- Consider specifying this operand if you want to retain the information in the performance trace information file for a long time. Estimate the number of performance trace information file generations and the file size from the following total capacity of the performance trace information files:

  Total capacity of the performance trace files = *pd_prf_file_count operand value* × *pd_prf_file_size operand value*

  Example

  This example specifies the total capacity of the performance trace information files required for retaining one hour worth of performance trace information.

pd_prf_level operand value	Total capacity of performance trace information files	
	Number of SQL statements executed per second is about 200	Number of SQL statements executed per second is about 500
`00000007`	750 MB (default value#)	750 MB (default value#)
`0000001f`	750 MB (default value#)	2 GB
`0000007f`	2 GB	5 GB

#: This is *default value of the pd_prf_file_count operand* × *default value of the pd_prf_file_size operand*.

- To approximate the total capacity of the performance trace information files required per unit, use the following guidelines:

  Total capacity of the performance trace information files = MAX(768,000, $a \times b \times c \times d$) (kilobytes)

  a: Number of front-end and back-end servers per unit

  For a HiRDB/Single Server or if there is only a system manager or a dictionary server, the value is 1.

  b: Number of SQL statements executed per second on each server

  You can obtain the average value per editing time by dividing `TOTAL`, the total SQL statement execution count in the SQL-related statistical information, by the editing time `EDIT TIME`.

  c: Average amount of performance trace collected per SQL statement (kilobytes)

  Assume one of the following values based on the value of the `pd_prf_level` operand:

  - `00000007`: 0.5 kilobytes
  - `0000001f`: 0.5 kilobytes
  - `0000007f`: 1 kilobyte

Note that the above value varies according to conditions, such as the type of SQL statement, amount of data, system configuration, whether facilities are applied, and tuning status.

d: Length of time for which the performance trace information is to be retained (seconds)

- The length of time for which the performance trace information can be retained per file might change drastically depending on numerous factors, such as the type of transaction and the HiRDB configuration. To obtain accurate values, measure the time for which performance trace information is retained per file when the most typical transactions and utilities are executed. You can determine the time value by using the timestamp or the `pdprfed` command's editing results. Calculate the required number of file generations based on the measured time.

Notes

- The value of this operand affects the amount of disk space required under the HiRDB directory. The following amount of disk space is required to store the performance trace information files:
  pd_prf_file_count operand value × *pd_prf_file_size operand value*
- This operand affects the capacity of the error information files that are collected by using commands, such as `pdinfoget`.
- If you change the value of this operand to a smaller value, the performance trace information files corresponding to the eliminated portion of the value will no longer be used, but they are not deleted automatically. If you need to delete any unneeded performance trace information file, delete them after HiRDB has terminated.

Relationship to other operands

This operand is related to the `pd_prf_file_size` operand.

119) pd_prf_file_size = *size-of-a-performance-trace-information-file*

~<unsigned integer> ((1024-1048576))<<51200>>(kilobytes)

Specifies the size of each performance trace information file.

Specification guidelines

Normally, this operand does not need to be specified. Consider specifying this operand if you want to retain the information in the performance trace information files for a longer than normal time.

Notes

- This operand value affects the amount of disk space required under the HiRDB directory. The following amount of disk space is required to store the performance trace information files:
  pd_prf_file_count operand value × *pd_prf_file_size operand value*
- This operand affects the capacity of the error information files that are collected by using commands, such as `pdinfoget`.
- If you increase this operand value, the time required by the `pdprfed` command to edit and output performance traces increases.
- If this operand value is too large, sending the files to the support service for purposes such as an investigation of performance issues might be difficult, even if the files are compressed. Specify the appropriate file size taking into account file transmission.
- The performance trace information is output to the performance trace information file every 10 seconds. If many traces are output at one time, the performance trace information file might become smaller in size by a maximum of 10 megabytes.

Relationship to other operands

This operand is related to the `pd_prf_file_count` operand.

2.2.24 Operands related to troubleshooting information

120) pd_cancel_dump = <u>put</u> | noput

This operand is designed to reduce the amount of troubleshooting information to be output.

Specifies whether troubleshooting information is to be collected in the following cases:

- When SQL does not terminate within the monitoring time specified by the `PDCWAITTIME` operand in the client environment definition

- When a UAP being executed is canceled by the `pdcancel` command

For details about the troubleshooting information to be collected, see the *HiRDB Version 9 System Operation Guide* and *Table 2-2 Error information that is displayed in the event of abnormal termination* in the section on the `pd_dump_suppress_watch_time` operand.

`put`:

Troubleshooting information is collected. Because the troubleshooting information is output to files under `%PDDIR%\spool`, a space shortage might occur in the file system.

Note that the troubleshooting information that has been collected is automatically deleted by HiRDB at the following timings.

- Every 24 hours while HiRDB is running (the deletion interval can be changed using the `pd_spool_cleanup_interval` operand).
- When HiRDB is started (whether to delete the troubleshooting information can be changed using the `pd_spool_cleanup` operand).

For the HiRDB administrator to delete troubleshooting information, the administrator must execute the `pdcspool` command.

`noput`:

Do not collect troubleshooting information. Because no troubleshooting information will be collected, the load on the file system is reduced. Specify this option if UAP cancellation occurs frequently during normal operation and there is no need to investigate the causes.

Troubleshooting information might be acquired even though `noput` is specified in this operand. For details, see *Table 2-2 Error information that is displayed in the event of abnormal termination* in the section on the `pd_dump_suppress_watch_time` operand.

121) pd_client_waittime_over_abort = Y | N

This operand is designed to reduce the amount of troubleshooting information to be output.

This operand specifies whether to collect the following types of troubleshooting information when the client maximum wait time (value of the `PDCWAITTIME` operand in the client environment definition) is exceeded during transaction execution.

- Shared memory dump (collected only once)
- Simple dump
- Detailed information inside a unit

Troubleshooting information is collected into `%PDDIR%\spool\save`.

`Y`:

Collect troubleshooting information. A shared memory dump is collected only when the maximum wait time of the first client after HiRDB activation is exceeded.

In the case of HiRDB/Parallel Server, troubleshooting information is collected from the back-end server processes and dictionary server processes that are related to the connected front-end server processes. In the case of HiRDB/Single Server, troubleshooting information is collected from the single server process.

`N`:

Do not collect troubleshooting information. In the case of a HiRDB/Parallel Server, troubleshooting information is not collected from the connected front-end server processes. Related back-end server processes and dictionary server processes are rolled back, and no troubleshooting information is collected from them. In the case of a HiRDB/Single Server, no troubleshooting information is collected from the single server process.

Notes

- If troubleshooting information is maintained indefinitely in the file, it will use up file space. Note that the troubleshooting information that has been collected is automatically deleted by HiRDB at the following timings.

  ● Every 24 hours while HiRDB is running (the deletion interval can be changed using the `pd_spool_cleanup_interval` operand).

  ● When HiRDB is started (whether to delete the troubleshooting information can be changed using the `pd_spool_cleanup` operand).

  For the HiRDB administrator to delete troubleshooting information, the administrator must execute the `pdcspool` command.

- Even when the client has exceeded its maximum wait time, it might not be possible to acquire troubleshooting information if the single server process or back-end server process that the client is connected to is in a critical state. Use the `pdls -d rpc` command to determine if a process is in a critical state.

Relationship to other operands

A HiRDB/Parallel Server can restrict the units that output shared memory dumps by specifying the `pd_clt_waittime_over_dump_level` operand, which can reduce the amount of output by shared memory dumps.

122) pd_clt_waittime_over_dump_level = <u>all</u> | shm_fesonly

This operand only applies to a HiRDB/Parallel Server. Its purpose is to reduces the amount of troubleshooting information that is output.

When `Y` (the default) is set in the `pd_client_waittime_over_abort` operand, the following troubleshooting information is acquired when the client's maximum wait time (value of the `PDCWAITTIME` operand in the client environment definition) is exceeded:

- Shared memory dump (acquired only once)
- Simple dump
- Detailed information from within the unit

In the case of shared memory dumps, which produce a large amount of output, switchover can occur due to the computer being overwhelmed or the computer might slow down because of the burden of the shared memory dump acquisition processing.

Specifying `shm_fesonly` in this operand enables the units permitted to output shared memory dumps to be limited, thus reducing the amount of shared memory dump output.

`all`:

All units are to be permitted to output shared memory dumps (no restrictions). (The amount of shared memory dump output will not be reduced.)

`shm_fesonly`:

The units permitted to output shared memory dumps are to be restricted. (The amount of shared memory dump output will be reduced.) Only units defined as front-end servers will be permitted to output a shared memory dump.

The troubleshooting information that is output depending on the specification of this operand is explained in the following table.

Type of troubleshooting information		FES unit connected to the client	Unit accessed by the transaction	Other units
Shared memory dump	`all` specified in this operand	Y	Y	Y
	`shm_fesonly` specified in this operand	Y	N	N
Simple dump		Y	Y	N
Detailed information from within the unit		Y	Y	Y

Legend:

Y: Troubleshooting information is acquired.

N: Troubleshooting information is not acquired.

Note

Simple dumps and detailed information from within a unit are acquired regardless of the specification of this operand.

Application criterion

Normally, this operand need not be specified.

When there are many units and multiple units are allocated to a single server machine, acquisition processing for shared memory dumps might be executed concurrently for several units on the same server machine.

If this happens, the overhead of shared memory dump acquisition processing might overwhelm the computer or slow it down, causing switchover to occur. In such a case, specify `shm_fesonly` to reduce the workload on the machine.

The minimum required troubleshooting information will always be acquired even when `shm_fesonly` is specified.

123) pd_dump_suppress_watch_time = *troubleshooting-information-output-suppression-time*

~<unsigned integer>((0-3600)) <<0>>(seconds)

This operand is designed to reduce the amount of troubleshooting information to be output.

This operand specifies the amount of time (in seconds) during which to suppress outputting the troubleshooting information again (files under `%PDDIR%\spool`) that is output when any of the following situations occurs.

- The time specified in `PDCWAITTIME` is exceeded.
- The UAP being executed is cancelled by the `pdcancel` command .
- A process terminates abnormally.

Once troubleshooting information is output, no troubleshooting information is output again until the time specified by this operand has elapsed. For example, if `60` is specified for this operand, no troubleshooting information is output again until 60 seconds have passed because troubleshooting information was previously output.

Note that if `0` is specified in this operand, outputting of troubleshooting information is not suppressed.

Advantage

When there are multiple HiRDB server processes, timing out, for example, might cause them to terminate abnormally one after the other. If abnormal terminations of server processes occur successively, troubleshooting information, such as core files and simple dumps, are repeatedly collected, thus causing a space shortage on the disk on which the HiRDB directory is located. If such a shortage occurs, HiRDB might terminate abnormally. Therefore, specify this operand to make sure that no disk space shortage occurs.

Notes

- When abnormal termination is caused by an internal conflict, or if a signal is received from outside, troubleshooting information is collected regardless of the value specified for this operand.
- The following table lists the error information that is displayed in the event of abnormal termination.

Table 2–2: Error information that is displayed in the event of abnormal termination

Cause of abnormal termination		Error information	pd_dump_suppress_watch_time value			
			0		Not 0	
			pd_cancel_dump value			
			put	noput	put	noput
`PDSWAITTIME` over		Save core file	N	N	N	N
		Snapshot of error	N	N	N	N
		Simple dump file	N	N	N	N
		`KFPA20009-W` message	N	N	N	N
		SQL runtime warning information file	N	N	N	N
`PDSWATCHTIME` over		Save core file	N	N	N	N
		Snapshot of error	N	N	N	N
		Simple dump file	N	N	N	N
		`KFPA20009-W` message	N	N	N	N
		SQL runtime warning information file	N	N	N	N
`PDCWAITTIME` over	`pd_client_waittime`	Save core file	Y	Y	Y+	Y+

Cause of abnormal termination		Error information	pd_dump_suppress_watch_time value: 0, pd_cancel_dump value: put	pd_dump_suppress_watch_time value: 0, pd_cancel_dump value: noput	pd_dump_suppress_watch_time value: Not 0, pd_cancel_dump value: put	pd_dump_suppress_watch_time value: Not 0, pd_cancel_dump value: noput
	`_over_abort=Y`	Snapshot of error	Y	Y	Y+	Y+
		Snapshot 2 of error	Y	Y	Y+	Y+
		Simple dump file	Y	Y	Y+	Y+
		`KFPA20009-W` message	Y	Y	Y+	Y+
		SQL runtime warning information file	Y	Y	Y+	Y+
		Shared memory dump file	F	F	F	F
	`pd_client_waittime` `_over_abort=N`	Save core file	N	N	N	N
		Snapshot of error	Y	N	Y+	N
		Snapshot 2 of error	N	N	N	N
		Simple dump file	Y	N	Y+	N
		`KFPA20009-W` message	Y	N	Y+	N
		SQL runtime warning information file	Y	N	Y+	N
		Shared memory dump file	N	N	N	N
`pdcancel` command		Save core file	N	N	N	N
		Snapshot of error	Y	N	Y+	N
		Simple dump file	Y	N	Y+	N
		`KFPA20009-W` message	Y	N	Y+	N
		SQL runtime warning information file	Y	N	Y+	N
Internal `kill9`#1		Save core file	N	N	N	N
		Snapshot of error	Y	N	Y+	N
		Simple dump file	Y	N	Y+	N
		`KFPA20009-W` message	Y	N	Y+	N
		SQL runtime warning information file	Y	N	Y+	N
Internal `kill3`#2		Save core file	Y	Y	Y+	Y+
		Snapshot of error	Y	Y	Y+	Y+
		Simple dump file	Y	Y	Y+	Y+
		`KFPA20009-W` message	Y	Y	Y+	Y+
		SQL runtime warning information file	Y	Y	Y+	Y+
Abort#3		Save core file	Y	Y	Y+	Y+
		Snapshot of error	Y	Y	Y+	Y+

Cause of abnormal termination	Error information	pd_dump_suppress_watch_time value			
		0		Not 0	
		pd_cancel_dump value			
		put	noput	put	noput
	Simple dump file	Y	Y	Y+	Y+
	`KFPA20009-W` message	Y	Y	Y+	Y+
	SQL runtime warning information file	Y	Y	Y+	Y+
	Abort information file	Y	Y	Y+	Y+
Other[#4]	Save core file	D	D	D	D
	Snapshot of error	Y	Y	Y	Y
	Simple dump file	D	D	D	D
	`KFPA20009-W` message	D	D	D	D
	SQL runtime warning information file	D	D	D	D

Legend:

Y: Outputs error information. The specification of the `pd_dump_suppress_watch_time` operand is invalid.

N: Does not output error information.

Y+: Outputs error information. The specification of the `pd_dump_suppress_watch_time` operand is valid.

D: Error information might not be output depending on how the process is terminated.

F: After the unit is started, error information is output during the first dump.

The units that output shared memory dumps can be restricted by specifying `shm_fesonly` in the `pd_clt_waittime_over_dump_level` operand.

#1: Issuance of internal `SIGKILL`, such as abnormal termination of a UAP by OpenTP1. Does not include `PDCWAITTIME` over or abnormal termination by the `pdcancel` command.

#2: Issuance of internal `SIGQUIT`, such as error detection. Does not include `PDCWAITTIME` over or abnormal termination by the `pdcancel` command.

#3: HiRDB detects a conflict and executes `abort()`.

#4: `SIGSEGV`, `SIGBUS`, receiving of a signal from outside, `exit`, and other unexpected errors.

124) pd_debug_info_netstat = Y | N

This operand is designed to reduce the amount of troubleshooting information to be output.

Specifies whether troubleshooting information (network information) is to be collected when a HiRDB process or HiRDB (unit) terminates abnormally.

`Y`: Collect network information.

`N`: Do not collect network information.

Troubleshooting information related to network information is output to the *server-name-n*`.deb` file under `%PDDIR%\spool\save,` where *n* is a serial number between 1 and 3. The following are file name examples:

Examples

sdsl.deb

fes2.deb

Specification guidelines

- Normally, this operand need not be specified.

- Specifying `N` reduces the load on the network associated with collecting network information when a HiRDB process or HiRDB (unit) terminates abnormally. However, it might not be possible to identify the causes of errors.

125) pd_spool_cleanup_interval = *troubleshooting-information-deletion-interval*

~<unsigned integer>((0-744))<<24>>(times)

This operand is used for deleting the troubleshooting information and temporary work files that have been output. If these items are not deleted, they might cause a space shortage on the disk on which the HiRDB directory is located. If such a shortage occurs, HiRDB might terminate abnormally. Therefore, HiRDB regularly deletes the following files:

- Troubleshooting information file (files in `%PDDIR%\spool`)
- Temporary work files (files in `%PDDIR%\tmp`)
- Files in the directory specified in the `pd_tmp_directory` operand

This operand specifies the deletion interval (hours). For example, if `48` is specified for this operand, these files are deleted every 48 hours. Normally (if this operand is omitted), files are deleted every 24 hours.

Note that time counting begins when HiRDB is normally started. When HiRDB is normally terminated, time counting also stops. Then, the count returns to 0 during the next normal startup.

Specify the files to be deleted using the `pd_spool_cleanup_interval_level` operand explained as follows.

Operand rule

If `0` is specified, files are not deleted.

Specification guidelines

If `24`, `48`, `72`, and so on are specified for this operand, files are deleted at the predetermined time. Specify the time so that files are deleted during the time period that does not overload the system.

Notes

Even while HiRDB is stopped because of planned termination, forced termination, or abnormal termination, time counting continues. However, if the deletion time arrives while HiRDB is stopped, files are not deleted. Files are not deleted until the next deletion time. To restart HiRDB after deleting the files, execute the `pdcspool` command.

Remarks

The difference between the `pd_spool_cleanup_interval` and `pd_spool_cleanup` operands is as follows:

- The `pd_spool_cleanup_interval` operand is related to regular deletion of troubleshooting information.
- The `pd_spool_cleanup` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup` operand.

126) pd_spool_cleanup_interval_level = *number-of-days* [, *deletion-type*]

This operand is used for deleting the troubleshooting information and temporary work files that have been output, and specifies the condition for regularly deleting the troubleshooting information and temporary work files.

***number-of-days:* ~<unsigned integer>((1-24855)) <<7>> (days)**

Troubleshooting information files that are older than the number of days specified here are deleted. For example, if `3` is specified, all troubleshooting information files, except for those created within the last 3 days (or 3 days × 24 hours = 72 hours), are deleted.

***deletion-type:* <character string> <<all>>**

Specifies the type of troubleshooting information file to be deleted.

`all`: All files are to be deleted.

`dump`: Only the files internally acquired by HiRDB are to be deleted.

The following are the types of troubleshooting information files that are deleted.

Troubleshooting information file type	Directory name	all	dump	Remarks
Deadlock and timeout information	`pdlckinf`	Y	N	Output when an error occurs during locking.
Access path information	`pdsqldump`	Y	N	Output when the access path display utility is used.
Save core file	`save`	Y	Y	Output when a process is abnormally terminated.
Shared memory dump file	`pdshmdump`	Y	Y	Output when a process or unit is abnormally terminated.
Simple dump file	`pdsysdump`	Y	Y	None
	`pdsdsdump`	Y	Y	Nonexistent in a HiRDB/Parallel Server
	`pdfesdump` `pddicdump` `pdbesdump`	Y	Y	Nonexistent in a HiRDB/Single Server
System log file status information file	`pdjnlinf`	Y	N	Files under `\pdjnlinf\errinf` are not deleted.

Y: File is deleted.

N: File is not deleted.

Note:

Directory names under `%PDDIR%\spool` are shown.

All temporary work files, except for those listed as follows, are deleted regardless of the deletion type specification. Parentheses indicate directory names under `%PDDIR%\tmp`.

- Current working directory (`home`) of the process in which HiRDB is to start
- Shared memory information file (`pdommenv`)
- Differential information files of the `pdbufls` command (files with names that begin with `CMb`)

Condition

A value other than `0` must be specified for the `pd_spool_cleanup_interval` operand.

Specification guidelines

- Specify a value that is longer than the execution time of commands (including utilities). For example, if the execution of the `pdcopy` command, which collects backup, requires 24 hours (1 day), specify at least 2 for the number of days. If you do not specify a value that is longer than the execution time of the command, the temporary work files being used by the command are deleted, and thus the command might not run correctly.
- If the specified value is too large, disk space might fill up; if the specified value is too small, information files needed for troubleshooting might be deleted.

Operand rule

If you specify a deletion type, you must also specify a number of days.

Note

If the `TMP` environment variable is specified but the `pd_tmp_directory` operand is not specified, temporary work files used by commands and utilities will be output to the directory specified in the `TMP` environment variable. The temporary work files that are output to the directory specified by the `TMP` environment variable are not subject to regular deletion. Therefore, use Windows Explorer, for example, to delete them.

Remarks

The difference between the `pd_spool_cleanup_interval_level` and `pd_spool_cleanup_level` operands is as follows:

- The `pd_spool_cleanup_interval_level` operand is related to regular deletion of troubleshooting information.

- The `pd_spool_cleanup_level` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval_level` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup_level` operand.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default deletion type for this operand is `dump`.

127) pd_spool_cleanup = normal | <u>force</u> | no

This operand is used for deleting the troubleshooting information that has been output.

Specifies whether troubleshooting information files (files under `%PDDIR%\spool`) that were output previously by HiRDB are to be deleted when HiRDB is started. This operand is related to the `pd_spool_cleanup_level` operand, described as follows.

`normal`:

Delete the files when HiRDB is started normally or is restarted following a planned termination.

`force`:

Delete the files whenever HiRDB is started, regardless of the HiRDB activation mode.

`no`:

Do not delete the files.

Specification guidelines

If troubleshooting information files take up too much disk space, specify `normal` or `force`.

Remarks

The difference between the `pd_spool_cleanup_interval` and `pd_spool_cleanup` operands is as follows:

- The `pd_spool_cleanup_interval` operand is related to regular deletion of troubleshooting information.
- The `pd_spool_cleanup` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup` operand.

128) pd_spool_cleanup_level = *number-of-days* [, *deletion-type*]

This operand is used for deleting the troubleshooting information that has been output, and specifies the condition for deleting the troubleshooting information files during HiRDB startup.

***number-of-days:* ~<unsigned integer>((0-24855)) <<7>> (days)**

Specifies a number of days when troubleshooting information that is older than the specified number of days is to be deleted. For example, if `3` is specified, all troubleshooting information will be deleted except for the information that is fewer than 3 days old (3 days × 24 hours = 72 hours).

If `0` is specified, all troubleshooting information files are deleted.

***deletion-type:* <character string> <<all>>**

Specifies the type of troubleshooting information to be deleted.

`all`: Delete all file types.

`dump`: Delete only files collected internally by HiRDB.

The following are the types of troubleshooting information files that are deleted:

Troubleshooting information file type	Directory name	all	dump	Remarks
Deadlock and timeout information	`pdlckinf`	Y	N	Output when an error occurs during locking.
Access path information	`pdsqldump`	Y	N	Output when the access path display utility is used.

Troubleshooting information file type	Directory name	all	dump	Remarks
Save core file	`save`	Y	Y	Output when a process is abnormally terminated.
Shared memory dump file	`pdshmdump`	Y	Y	Output when a process or unit is abnormally terminated.
Simple dump files	`pdsysdump`	Y	Y	None
	`pdsdsdump`	Y	Y	Nonexistent in a HiRDB/Parallel Server
	`pdfesdump` `pddicdump` `pdbesdump`	Y	Y	Nonexistent in a HiRDB/Single Server
System log file status information file	`pdjnlinf`	Y	N	Files under `\pdjnlinf\errinf` are not deleted.

Y: File is deleted.

N: File is not deleted.

Note: Directory names under `%PDDIR%\spool` are shown.

Condition

`normal` or `force` (default value) must be specified for the `pd_spool_cleanup` operand.

Operand rule

A number of days and a deletion type must both be specified.

Remarks

The difference between the `pd_spool_cleanup_interval_level` and `pd_spool_cleanup_level` operands is as follows:

- The `pd_spool_cleanup_interval_level` operand is related to regular deletion of troubleshooting information.
- The `pd_spool_cleanup_level` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval_level` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup_level` operand.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default deletion type for this operand is `dump`.

129) pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*

~<unsigned integer>((126-16383))<<126>>

A HiRDB process records the history of the executed functions and macros inside the process private memory. This history is called a *module trace*. This operand specifies the number of module trace records. The content of this history is loaded into the `core` file and is output when a process error occurs.

Specification guideline

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Note

Process private memory of the following size is allocated to each process:

In the 32-bit mode: 64 + 48 × *pd_module_trace_max operand value* (bytes)

In the 64-bit mode: 64 + 64 × *pd_module_trace_max operand value* (bytes)

130) pd_module_trace_timer_level = <u>0</u> | 10 | 20

Specifies how to acquire the time to be output in module traces. The following table explains the meaning of the value specified for this operand.

Specified value	Time acquisition method
`0`	Time is output in seconds at every module trace output location.
`10`	Time is output in microseconds only at performance-critical module trace output locations, such as those before and after input/output processing, and time is output in seconds at other locations.
`20`	Time is output in microseconds at every module trace output location.

Specification guideline

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Note

If you specify a value other than `0` for this operand, a function for acquiring time in microseconds is issued, and as a result, system performance might decline.

131) pd_pth_trace_max = *maximum-number-of-stored-communication-traces*

~<unsigned integer>((1024-8388608))<<1024>>

Specifies the maximum number of communication trace records to be used as troubleshooting information.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reason such as performance checking, follow the maintenance engineer's instructions.

Notes

Increasing the value of this operand increases the amount of process private memory secured by HiRDB processes.

Process private memory for communication traces is calculated based on this operand's value rounded up to the power of two. For details about memory requirements, see *Calculation of required memory* in the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_pth_trace_max` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

2.2.25 Operands related to RDAREAs

132) pd_max_rdarea_no = *maximum-number-of-RDAREAs*

~<unsigned integer>((5-8388592)) <<200>>

Specifies the maximum number of RDAREAs allowed. If the total number of RDAREAs exceeds the value specified for this operand, HiRDB cannot be started normally. Here, RDAREAs also include master directory RDAREAs, data directory RDAREAs, and data dictionary RDAREAs.

Specification guidelines

- Specify a value that is equal to or greater than the total number of RDAREAs (leave some extra room). You can use the `pddbls` command to check the total number of RDAREAs. If you plan to add RDAREAs, specify a value by taking the additional RDAREAs into account.
- For a HiRDB/Parallel Server, the value of this operand is applied to each back-end server. For example, if `100` is specified for this operand, a maximum of 100 RDAREAs can be created in each back-end server. Therefore, use the largest number of RDAREAs among the back-end servers as the guideline when specifying a value for this operand.
- If you are using shared RDAREAs, also add the number of shared RDAREAs used by reference-only back-end servers.

Notes

- Do not specify an unnecessarily large value for this operand. Increasing the specification value of this operand increases the size of the shared memory used by HiRDB. If a shortage occurs in the shared memory, HiRDB might not be able to start.

Effects on individual estimation formulas

If the value of the `pd_max_rdarea_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*
- *Formula 1*, *Formula 2*, and *Formula 3* under *Formulas for shared memory used by a single server*
- *Formula 1* and *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*
- *Memory size required for the execution of the database structure modification utility (pdmod)*
- *Formula 1*, *Formula 2*, and *Formula 3* under *Formulas for the size of the shared memory used by a back-end server*
- *Formula for the shared memory used by a front-end server*

133) pd_max_file_no = ***maximum-number-of-HiRDB-files-comprising-an-RDAREA***

~<unsigned integer>((5-134217728)) <<400>>

Specifies the maximum number of HiRDB files that comprise an RDAREA. If the total number of HiRDB files exceeds the value specified for this operand, HiRDB cannot be started normally. Here, HiRDB files comprising an RDAREA also include the HiRDB files of master directory RDAREAs, data directory RDAREAs, and data dictionary RDAREAs.

Specification guidelines

- Specify a value that is equal to or greater than the total number of HiRDB files comprising an RDAREA (leave some extra room). You can use the `pdfstatfs` command to check the number of HiRDB files inside each HiRDB file system area. If you plan to add HiRDB files, specify a value by taking them into account. HiRDB files are added when RDAREAs are added, reinitialized, or extended.
- For a HiRDB/Parallel Server, the value of this operand is applied to each back-end server. For example, if `100` is specified for this operand, a maximum of 100 HiRDB files can be created in each back-end server. Therefore, use the largest number of HiRDB files among the back-end servers as the guideline when specifying a value for this operand.
- If you are using shared RDAREAs, also add the number of HiRDB files comprising the shared RDAREAs used by reference-only back-end servers.

Notes

- Do not specify an unnecessarily large value for this operand. Increasing the specification value of this operand increases the size of the shared memory used by HiRDB. If a shortage occurs in the shared memory, HiRDB might not be able to start.

Effects on individual estimation formulas

If the value of the `pd_max_file_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 2* under *Formulas for shared memory used by a single server*
- *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*

- *Formula 2* under *Formulas for the size of the shared memory used by a back-end server*

134) pd_rdarea_warning_point = ***segment-usage-ratio-1*****[,*****segment-usage-ratio-2*****[,*****segment-usage-ratio-3*****]]**

~<unsigned integer>((0-100))(%)

Specifies triggers for issuing warning messages concerning RDAREA segment usage and area usage by HiRDB files that can be extended. Each trigger is specified as a percentage. The warning messages are `KFPH00211-I` or `KFPA12300-I` for RDAREA segment area usage, and `KFPH22037-W` for area usage by HiRDB files that can be extended.

When auto-extension of RDAREAs is not used

The `KFPH00211-I` or `KFPA12300-I` warning message about segment usage is issued.

The following table shows the relationship between specification of this operand and the segment usage ratio warning message for the different types of RDAREAs.

RDAREA type	pd_rdarea_warning_point not specified	pd_rdarea_warning_point specified
• Master directory RDAREA • Data directory RDAREA	No message is output.	Segment use has started that makes the segment usage ratio equal to the value specified for all segments of the applicable RDAREA.
• Data dictionary RDAREA • User RDAREA	Segment use has started that makes the segment usage ratio 80%, 90%, or 100% in terms of the relative position of the segment in the last file of the applicable RDAREA.	Segment use has started that makes the segment usage ratio equal to the value specified for all segments of the applicable RDAREA.
• LOB RDAREA	Segment use has started that makes the segment usage ratio 80%, 90%, or 100% in terms of the relative position of the segment in the last file of the applicable RDAREA.	Use of the segment corresponding to the specified relative position has started.

When auto-extension of RDAREAs is used

The `KFPH22037-W` warning message about area usage by HiRDB files that will be extended automatically is issued.

When RDAREA auto-extension is being used, the message output depends on whether this operand is specified:

Operand specification	pd_rdarea_warning_point omitted	pd_rdarea_warning_point specified
Message output condition	The extended HiRDB file area allocated exceeds a usage ratio of 80%, 90%, or 100%.	The extended HiRDB file area allocated exceeds a specified usage ratio.

The HiRDB file area usage that will be extended automatically is the percentage used of the maximum area that can be added. It is determined as follows.

Area usage by file that will be extended automatically (%)=A $\div$ *B* $\times$ 100

- For a HiRDB file system area specified by the `pdfmkfs -e` command or a HiRDB file system area for disk access that is specified by `pdfmkfs -a`:

  A: Capacity of HiRDB file that will be extended automatically

  B: Maximum capacity that can be extended automatically = min {(*A* + *C*), 64 GB}

  C: Free space in the HiRDB file system that contains the HiRDB file that will be extended automatically

- For a HiRDB file system area in the OS's file system that is specified by the `pdfmkfs -a` command:

  A: Capacity of HiRDB file that will be extended automatically

  B: Maximum capacity that can be extended automatically = min {(*A* + *C*), 64 GB}

  C: Free space in the HiRDB file system that contains the HiRDB file that will be extended automatically + Free space on the disk where the HiRDB file system that contains the HiRDB file that will be extended automatically is located

If the number of used extents of the automatically extended HiRDB file is 20, 22, 24, or higher, a warning message (`KFPH22038-W`) for the number of used extents of the HiRDB file that will be extended automatically will be displayed.

The following table shows specification examples of this operand:

Message output condition	Operand specification value
To output a message when segment use has started that makes the segment usage ratio 80%, 90%, or 100% of all the segments in the applicable RDAREA	`pd_rdarea_warning_point = 80,90,100`
To output a message when segment use has started that makes the segment usage ratio 50% or 90% of all the segments in the applicable RDAREA	`pd_rdarea_warning_point = 50,90`
To not output a segment usage warning message	`pd_rdarea_warning_point = 0`

Operand rules

- Up to three values can be specified.
- When the same value is specified more than once, only one message will be output for that value.
- When `0` is specified for all three values, no segment usage warning message is output.
- When `0` and a non-zero numeric value are both specified, the non-zero numeric value is used as the trigger for message output.
- The `KFPA12300-I` segment usage warning message is issued only if `YES` is specified in the `PDEXWARN` client environment definition.

Note

Specification of this operand is not applicable to the database initialization utility (`pdinit`) or the database recovery utility (`pdrstr`).The message content is the same regardless of whether this operand is specified.

135) pd_rdarea_warning_point_msgout = Y | N

Specifies whether an RDAREA segment usage warning message (`KFPH00211-I` or `KFPA12300-I`) is to be issued.

`Y`: Issues an RDAREA segment usage warning message according to the specification of `pd_rdarea_warning_point`.

`N`: Does not issue an RDAREA segment usage warning message.

Specification guidelines

For guidance about the RDAREA segment usage warning message specifications, see *Environment settings* under *Free space reusage facility* in the *HiRDB Version 9 Installation and Design Guide*.

Notes

In the following cases, a segment usage warning message can be issued, even though `N` is specified in this operand.

- When data is being stored by the database creation utility (`pdload`) or the database organizing utility (`pdrorg`)
- When data is being stored in tables that do not use the free space reusage facility
- When data is being stored in an index
- When configuration columns of tables that use the free space reusage facility include variable-length columns, and updates that increase the data size occur frequently
- When the free space reusage facility can no longer be used because the number of table definitions has exceeded the value specified in the `pd_assurance_table_no` operand

Relationship to other operands

This operand is related to the following operands:

- `pd_rdarea_warning_point`
- `pd_assurance_table_no`

136) pd_rdarea_extension_timing = use | nouse

Specifies the timing for performing RDAREA automatic extension when the RDAREA auto-extension facility is being used.

`use`:

Extend automatically the number of free segments in an RDAREA when the number of available free segments becomes equal to or less than the number of segments to be extended in a single auto-extension. For example, when the number of segments extended in a single auto-extension is 50, auto-extension will occur whenever the number of free segments drops to 50 or fewer.

`nouse`:

Extend automatically only when there are no free segments left in an RDAREA and new segments cannot be secured.

Specification guidelines

Advantages and disadvantages of the specification options are described below.

Item	Operand specification	
	use	nouse
Advantage	Empty segments (segments in the extended portion) can be used until addition of data causes an insufficient-pages error, even if auto-extension cannot be performed.# During this period, the RDAREA's status can be analyzed and reorganized to reflect its use status, the RDAREA can also be expanded, thereby minimizing the impact on operations.	Because auto-extension is performed only after there are no more empty segments, storage efficiency is improved when multiple HiRDB files are being created in the HiRDB file system area.
Disadvantage	Storage efficiency might be degraded when multiple HiRDB files are being created in the HiRDB file system area.	If auto-extension cannot be performed,# there are no empty segments, and operations might stop if the addition of data causes an insufficient-pages error.

#: Auto-extension cannot be used in the following cases:

- The mode that expands HiRDB files in the HiRDB file system area (`pdfmkfs -e` command) is being used, but there is no free space in the HiRDB file system area.
- The mode that expands the HiRDB file system area automatically (`pdfmkfs -a` command) is being used, but there is no free disk space or the HiRDB files have reached their maximum size (64 GB).
- Auto-extension has stopped because RDAREA backup has shut down.

We recommend that you specify `use` in this operand to minimize the impact on operations.

137) pd_rdarea_expand_format = Y | <u>N</u>

Specifies whether to initialize the extended area when an RDAREA is extended automatically.This operand can be specified for HiRDB file system areas in the OS's file system that are extended through specification of the `-a` option in the `pdfmkfs` command.

`Y`: Initializes the extended area.

`N`: Does not initialize the extended area

Specification guidelines

When `Y` is specified in this operand, disk I/Os occur for initializing the extended area. Because such disk I/O processing requires time, the following transactions can be impacted by delays:

- Transactions that update segments being initialized
- Transactions that allocate new segments in the extended area

We recommend that you specify `N` for this operand in most environments.

However, this recommendation depends on the specifications for allowing monitoring of capacity in RDAREAs and other locations. For details, see *Applying RDAREA automatic extension* in the *HiRDB Version 9 System Operation Guide*.

Note

When `Y` is specified in this operand, updating transactions that access an RDAREA being auto-extended might take longer to execute, which means that transactions might also be canceled by one of the following factors:

- Lock release wait time is exceeded
- HiRDB client maximum wait time or maximum utility execution time is exceeded

To avoid such types of transaction cancellation, estimate in seconds (as shown in the guide below) the processing time that will be needed for initialization of extended areas, considering the RDAREA configuration information, the I/O performance of the disk where the HiRDB file system area is allocated, and the operands listed in *Relationship to other operands*.

- Page size
- Segment size
- Extended segment count

Guide to processing time requirement for initialization of extended areas (sec) =
page-size (bytes) $\times$ (*segment-size* $\times$ *extended-segment-count* + α) $\div$ *I/O-performance-of-disk-where-HiRDB-file-system-area-is-allocated* (bytes/sec)

α : Page count of the creating directory at time of extension

- For data dictionary, user, or registry RDAREAs
  $\uparrow d \div b \uparrow + \uparrow d \div f \uparrow$
- For LOB RDAREAs
  $\uparrow S \div 64{,}000 \uparrow \times 96$

d: Extended segment count
b: $\downarrow (P - 20) \div (\uparrow S \div 32 \uparrow \times 8 + 56) \downarrow$
f: $\downarrow (125 \times P) \div (16 \times b) \downarrow \times b$
P: Page size
S: Segment size

Relationship to other operands

This operand is related to the following operands:

- `PDCWAITTIME`
- `pd_lck_wait_timeout`
- `pd_utl_exec_time`

138) pd_rdarea_open_attribute_use = Y | <u>N</u>

Specifies whether to use the `DEFER` or `SCHEDULE` attribute as the RDAREA opening trigger.

`Y`: Uses the `DEFER` or `SCHEDULE` attribute.

`N`: Does not use the `DEFER` or `SCHEDULE` attribute.

When this operand is omitted or when `N` is specified, the RDAREA opening trigger attribute is always `INITIAL`. Therefore, even if the `DEFER` or `SCHEDULE` attribute is specified as the RDAREA opening trigger in the operand or utility described as follows, the specification is invalid.

- `pd_rdarea_open_attribute`
- Database initialization utility
- Database structure modification utility

Notes

- When `Y` is specified, HiRDB requires a larger shared memory. Consequently, a shared memory shortage might occur, preventing the HiRDB system from starting.
- If the rapid system switchover facility, standby-less system switchover (1:1) facility or standby-less system switchover (effects distributed) facility is used, `Y` is assumed for this operand. Because the size of the shared memory used by the server increases as a result, re-estimate the shared memory size. For the formula for estimating the size of the shared memory used by a server, see the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_rdarea_open_attribute_use` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Processes started by HiRDB/Parallel Server*

- *Formulas for shared memory used by a single server*
- *Formulas for the size of the shared memory used by a back-end server*

139) pd_rdarea_open_attribute = INITIAL | DEFER | SCHEDULE

Specifies the standard value for the RDAREA opening trigger attribute.

The attribute specified by this operand is assumed for RDAREAs for which `open attribute` is not specified by the database initialization utility or the database structure modification utility.

For System RDAREAs, `INITIAL` is always assumed.

Specification guidelines

- Specify the same attribute for the RDAREAs in the same HiRDB file system area. If different attributes are specified, the expected result might not be obtained.
- The following table lists the opening/closing triggers for each attribute and their advantages and disadvantages.

Attribute	Initial status	Opening trigger	Closing trigger	Advantage	Disadvantage
`INITIAL`	Open	• HiRDB startup • `pdopen` command execution	`pdclose` command execution	Fast execution from the first SQL.	System startup takes time.
`DEFER`	Closed	• Initial access to RDAREA • `pdopen` command execution	`pdclose` command execution	• Fast system startup. • Regular SQLs are executed at high-speed, as well as after the initial access.	First access to each RDAREA takes time.
`SCHEDULE`	Closed	• Initial RDAREA access inside transaction • `pdopen` command execution	• Transaction termination • `pdclose` command execution	• Fast system startup. • Avoids concentrated file opening.	Initial access to RDAREA results in high workload for a transaction.

- The following table shows the operation mode appropriate to each attribute:

Attribute	Appropriate operation mode
`INITIAL`	HiRDB file system area is opened at system startup and keeps the RDAREA information resident in the memory. The HiRDB file system area is opened also during the initial RDAREA access. Because the RDAREA information is not re-created in this case, high-speed operation can be performed from the first SQL. The initial status of the RDAREA at system startup is open, and this RDAREA status will not change unless an operation command is entered subsequently, except during shift to an error shutdown. *This attribute is recommended unless an unusual operation mode is used.* When this attribute is used, a closed RDAREA cannot be accessed.
`DEFER`	HiRDB file system area is not opened at system startup. Instead, the HiRDB file system area is opened during the initial RDAREA access and the RDAREA information is kept resident in the memory. In the second and subsequent accesses, the processing beyond opening of the HiRDB file system area is not performed, and thus high-speed operations can be achieved. The initial status of the RDAREA at system startup is closed, and each RDAREA is opened during the initial access to that RDAREA. The RDAREA status will not change unless an operation command is entered subsequently, except during a shift to an error shutdown. Specify this attribute if you want to avoid cases in which a large number of HiRDB file system areas are opened concurrently or if you want to shorten the time required for starting HiRDB. When HiRDB is restarted, the RDAREAs to be recovered are opened during recovery processing. When this attribute is used, a closed RDAREA can also be accessed.

Attribute	Appropriate operation mode
`SCHEDULE`	HiRDB file system area is not opened at system startup. Instead, after HiRDB startup, the HiRDB file system area is opened during the initial RDAREA access in each transaction and the RDAREA information is kept resident in the memory. When a transaction terminates, the HiRDB file system area opened in that transaction is closed. Thereafter, the processing beyond opening will also be performed during the initial access to an RDAREA whenever the transaction is changed, and thus the workload required for transactions will increase. The initial status of the RDAREA at system startup is closed, and the RDAREA is kept open only during the transaction for the accessed RDAREA. When a transaction terminates, all RDAREAs that were opened in the transaction are closed. If the `pdopen` command is entered, these RDAREAs can be kept open until they are closed by the next shutdown. It is also possible to use other operation commands to change the status of the RDAREAs. If an error is detected, an error shutdown occurs. *This attribute is recommended when it is necessary to compensate for many HiRDB file system areas being opened simultaneously, or when it is necessary to reduce the HiRDB system startup time.* When HiRDB is restarted, the RDAREAs to be recovered are opened during recovery processing and closed after the completion of the recovery processing. When this attribute is used, a closed RDAREA can also be accessed.

Note

The following table describes the notes related to the use of the rapid system switchover facility, standby-less system switchover (1:1) facility, and standby-less system switchover (effects distributed) facility.

Facility used	Notes
Rapid system switchover facility	A standby unit that is targeted by the rapid system switchover facility has not opened RDAREAs while it is in a standby state. Furthermore, to minimize the time required for system switchover, the standby unit opens only those RDAREAs that are necessary for full recovery when system switchover occurs, and does not open other RDAREAs. Therefore, the RDAREA opening trigger for the standby system cannot be `INITIAL`. The `INITIAL` attribute of RDAREAs is changed to `DEFER`.
Standby-less system switchover (1:1) facility	To minimize the time required for system switchover, the standby-less system switchover (1:1) facility opens only those RDAREAs that are necessary for full recovery when system switchover occurs, and does not open other RDAREAs. Therefore, the opening trigger for the RDAREAs in the normal BES or alternate portion is as follows: • When system switchover occurs, the opening trigger for the RDAREAs in the alternate portion is `SCHEDULE`. • When the error is corrected and the system switches back to the normal BES, the opening trigger for RDAREAs with the `INITIAL` or `DEFER` attribute under the normal BES is changed to `DEFER`. RDAREAs with the `SCHEDULE` attribute retain this attribute.
Standby-less system switchover (effects distributed) facility	To minimize the time required for system switchover, the standby-less system switchover (effects distributed) facility opens only those RDAREAs that are necessary for full recovery when system switchover occurs, and does not open other RDAREAs. Therefore, when system switchover occurs, the opening trigger for RDAREAs with the `INITIAL` attribute under the guest BES is changed to `DEFER`.

The following table shows the relationships between the opening trigger for RDAREAs and the rapid system switchover facility, standby-less system switchover (1:1) facility, and standby-less system switchover (effects distributed) facility.

Condition		pd_rdarea_open_attribute_use specification value			
		N	Y		
			RDAREA opening trigger		
			INITIAL	DEFER	SCHEDULE
System switchover facility not used#1		`INITIAL`	`INITIAL`	`DEFER`	`SCHEDULE`
Standby system switchover facility#1	Rapid system switchover facility not used				

Condition			pd_rdarea_open_attribute_use specification value			
			N	Y		
				RDAREA opening trigger		
				INITIAL	DEFER	SCHEDULE
	Rapid system switchover facility	Running system				
		Standby system	`DEFER`[#2]	`DEFER`	`DEFER`	`SCHEDULE`
Standby-less system switchover (1:1) facility[#1]	Accepting portion (other than the alternate portion of the alternate BES unit)	Running system	`INITIAL`[#2, #3]	`INITIAL`[#3]	`DEFER`	`SCHEDULE`
		Standby system	`DEFER`[#2]	`DEFER`	`DEFER`	`SCHEDULE`
	Alternate portion	Running system	`SCHEDULE`[#2]	`SCHEDULE`	`SCHEDULE`	`SCHEDULE`
		Standby system				
Standby-less system switchover (effects distributed) facility[#1]	Normal start or restart		`INITIAL`	`INITIAL`	`DEFER`	`SCHEDULE`
	Restart due to system switchover		`DEFER`[#2]	`DEFER`	`DEFER`	`SCHEDULE`

#1: The opening trigger for the System RDAREA is `INITIAL`.

#2: It is assumed that `pd_rdarea_open_attribute_use` = `Y` is specified.

#3: Changed to `DEFER` after a restart.

140) pd_shared_rdarea_use = Y | <u>N</u>

This operand is applicable only to HiRDB/Parallel Server.

Specifies whether to use a shared RDAREA.

`Y`: A shared RDAREA is used.

`N`: A shared RDAREA is not used.

Notes

- If you omit this operand (or specify `N` for it) even though a shared RDAREA has been defined, HiRDB cannot be normally started.
- Specifying `Y` for this operand allocates the same shared memory block as that allocated when `commit` is specified for the `pd_dbsync_point` operand.

Effects on individual estimation formulas

If the value of the `pd_shared_rdarea_use` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for the size of the shared memory used by a back-end server*

141) pd_db_access_error_action = <u>dbhold</u> | unitdown

Specifies the action to be taken when a file access error occurs while accessing an RDAREA.

If an RDAREA is being accessed, and a process other than HiRDB has opened with an exclusive lock a file in the same HiRDB file system area that constitutes the RDAREA being accessed, a file access error results[#1] (error code `-1540`[#2]). This operand specifies the action to be taken in response to such a file access error.

#1: If the RDAREA is accessed while the HiRDB file system area is being copied using Windows `COPY` command, a file access error might occur. For the master directory RDAREA, the unit is shut down without resulting in the RDAREA in error shutdown status, even if the facility for taking a unit down when a file access error is detected is not used. When the unit is shut down, the `KFPH23040-I` message is issued.

#2: This error code is also returned when a mistake is made in setting the HiRDB file system area access authority.

`dbhold`:

Shut down the RDAREA being accessed when a file access error occurs.

`unitdown`:

Issue the `KFPH23040-I` message when a file access error occurs, and then apply the *facility for taking a unit down when a file access error is detected*. For details about the facility for taking a unit down when a file access error is detected, see the *HiRDB Version 9 System Operation Guide*.

Application criterion

Normally, this operand need not be specified.

Consider specifying this operand if a process other than HiRDB might perform an operation that accesses a HiRDB file system area made up of RDAREAs while HiRDB is operating, and that process is able to obtain an exclusive lock on the HiRDB file system area.

Notes

- When a file access error occurs and `unitdown` has been specified, the RDAREA being processed might be subject to error shutdown in the following cases:
  - ● A UAP or utility is being executed in the pre-update log acquisition mode or the no-log mode.
  - ● A UAP or utility is being run in a user LOB RDAREA that has been placed in the no-log mode by specification of `NO` in the `RECOVERY` operand of `CREATE TABLE`.

  If you use the facility for taking a unit down when a physical error is detected, avoid running these operations, if possible. If you need to run these operations, make a backup prior to running the UAP or utility so that recovery from an RDAREA error shutdown can be performed. For details about making back-ups, see the *HiRDB Version 9 System Operation Guide*.
- During recovery processing by the database recovery utility (`pdrstr`), the unit will not be shut down, even if `unitdown` is specified. In this case, re-execute `pdrstr` to recover.

Relationship to other operands

- `pd_mode_conf` operand
- `pd_db_io_error_action` operand
- `pd_db_hold_action` operand

If `unitdown` is specified in more than one of the `pd_db_io_error_action`, `pd_db_access_error_action`, and `pd_db_hold_action` operands, the operand value that takes effect is determined in the following order:

1. `pd_db_io_error_action` operand
2. `pd_db_access_error_action` operand
3. `pd_db_hold_action` operand

If more than one RDAREA input/output, file access, or physical error has occurred, determine the error that caused unitdown based on the above priority. In addition, see the message that is issued.

142) pd_db_hold_action = <u>dbhold</u> | unitdown

Specifies the action to be taken when a physical error (that results in shutdown with `I/O error occurred` or `open error occurred` displayed as the reason in the `KFPH00306-E` message) occurs while accessing an RDAREA. If this is the master directory RDAREA, a unitdown, not error shutdown, occurs, even if the facility for taking a unit down when a physical error is detected is not used. For details about the facility for taking a unit down when a physical error is detected, see the *HiRDB Version 9 System Operation Guide*.

`dbhold`:

Shut down the RDAREA being accessed when a physical error occurs.

`unitdown`:

Issue the `KFPH23047-I` message when a physical error occurs, and then use the facility for taking a unit down when a physical error is detected.

Application criterion

Normally, this operand does not need to be specified.

Consider specifying this operand if your system has a small number of RDAREAs and the entire application is shut down if one of the RDAREAs is placed in error shutdown status.

If you specify `unitdown`, also specify `MANUAL2` in the `pd_mode_conf` operand.

Notes

- If a physical error occurs while accessing an RDAREA and the `KFPH00307-E` is output, resulting in the RDAREA being placed in command hold status, the unit will not be shut down event if `unitdown` is specified in the `pd_db_hold_action` operand.
- If a physical error occurs and `unitdown` has been specified, the RDAREA being processed might be subject to error shutdown in the following cases:
  ● A UAP or utility is being executed in the pre-update log acquisition mode or the no-log mode.
  ● A UAP or utility is being run in a user LOB RDAREA that has been placed in the no-log mode by specification of `NO` in the `RECOVERY` operand of `CREATE TABLE`.

  If you use the facility for taking a unit down when a physical error is detected, avoid running these operations, if possible. If you need to run these operations, make a backup prior to running the UAP or utility so that recovery from an RDAREA error shutdown can be performed. For details about making back-ups, see the *HiRDB Version 9 System Operation Guide*.
- During recovery processing by the database recovery utility (`pdrstr`), the unit will not be shut down, even if `unitdown` is specified. In this case, re-execute `pdrstr` to recover.

Relationship to other operands

This operand is related to the following operands:

- `pd_mode_conf` operand
- `pd_db_io_error_action` operand
- `pd_db_access_error_action` operand

If `unitdown` is specified in more than one of the `pd_db_io_error_action`, `pd_db_access_error_action`, and `pd_db_hold_action` operands, the operand value that takes effect is determined in the following order:

1. `pd_db_io_error_action` operand
2. `pd_db_access_error_action` operand
3. `pd_db_hold_action` operand

If more than one RDAREA input/output, file access, or physical error has occurred, determine the error that caused unitdown based on the above priority. In addition, see the message that is issued.

2.2.26 Operands related to global buffers

143) pd_dbbuff_lru_option = SEPARATE | <u>MIX</u>

Specifies the LRU management method to be applied to global buffers. Note that `MIX` is assumed unconditionally when either of the following applies:

- `commit` is specified in `pd_dbsync_point`
- `N` is specified in `pd_dbbuff_binary_data_lru`

`SEPARATE`:

Manage the reference and update buffers with separate LRUs. If a global buffer shortage occurs, the reference buffer in the global buffer that was accessed first is purged from the memory. Specify this option when the number of references and updates per transaction is relatively small, as in an online job.

`MIX`:

Manage all global buffers with a single LRU. If a global buffer shortage occurs, the buffer in the global buffer that was accessed first is purged from the memory.

Specify this option when both a large number of retrievals and a large number of updates occur, as when an online job and a batch job coexist.

For details about the LRU management method for global buffers, see the manual *HiRDB Version 9 Description*.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `SEPARATE`.

144) pd_dbbuff_binary_data_lru = Y | N

Specifies whether LRU management is to be used for global buffers when executing a UAP that accesses a column for which BINARY type was specified. For details about setting the UAP to suppress LRU management of binary data to be accessed, see the *HiRDB Version 9 Installation and Design Guide*.

`Y`:

Applies LRU management to the branch row page at which binary data is stored and caches the page in a global buffer.

`N`:

Does not apply LRU management to the branch row page at which binary data is stored, but caches the page in a global buffer as the least recently accessed page, regardless of access frequency.

However, applies LRU management to the base row page and caches it in a global buffer.

When a UAP that accesses large amounts of large binary data using a global buffer is executed, the binary data is cached in the global buffer, so the contents most recently cached in the global buffer are pushed from memory, which can temporarily degrade performance. In such a case, if access frequency is low for binary data of a large size, this performance decline can be avoided by suppressing LRU management.

The table below describes LRU management and the advantages and disadvantages of each specification.

Item	Operand specification	
	Y	N
LRU management	When all global buffers are managed by the LRU method and there are insufficient global buffers, the least recently accessed global buffer in the global buffer pool is pushed from memory.	LRU management of global buffer is suppressed for branch row pages that store binary data and the least recently accessed page is cached. When global buffers are insufficient, the global buffer of the branch row page that stores the binary data is pushed from memory.
Advantage	Global buffers are used evenly, regardless of the type of access data, so UAP performance is balanced.	Binary data is pushed first from the global buffer, even if accessed after large binary data, so search performance is maintained for non-binary data.
Disadvantage	When accessing after large binary data, the most recently accessed data is pushed from the global buffer, so search performance declines for non-binary data.	• The buffer hit ratio declines in UAPs that access large binary data. That might increase the number of I/Os and decrease response performance. • When binary data is updated, the system log volume increases with frequent pushing out of binary-data branch rows.

Specification guidelines

- Normally, this operand is not specified.
- For systems that only accumulate without accessing binary data, specify `N`. For systems that frequently access binary data, specifying `N` in this operand has major disadvantages, so consider carefully before using it.
- For systems that have tables that include the `BINARY` type, abstract data types that include `BINARY` type attributes, and large binary data (such as `XML` type data), specify `N` for this operand if there is almost no access to the large binary data. Because binary data is preferentially pushed from the global buffer regardless of how frequently a stored page is accessed, the frequency with which pages that store non-binary data are pushed out of the global buffer can be minimized.

Relationship to client environment definition

When `NO` is specified in `PDDBBUFLRU` in the client environment definition, LRU management is suppressed for all data that the specified UAP will access, regardless of the specification of this operand.

Relationship to other operands

If you specify `N` for this operand, `MIX` is assumed for the `pd_dbbuff_lru_option` operand, even if `SEPARATE` is specified in `pd_dbbuff_lru_option`.

145) pd_dbbuff_modify = Y | N

Specifies whether the global buffer is dynamically modified by the `pdbufmod` command while HiRDB is running. For details about dynamic modification of the global buffer, see the *HiRDB Version 9 System Operation Guide*.

`Y`: Dynamically modifies the global buffer.

`N`: Does not dynamically modify the global buffer.

Condition

If `Y` is specified for this operand (to dynamically modify the global buffer), HiRDB Advanced High Availability is required.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_no`
- `pd_max_add_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_dbbuff_modify` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*
- *Formula 2*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for shared memory used by a single server*
- *Formula 2*, *Formula 3*, *Formula 4*, and *Formula 5* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 2*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for the size of the shared memory used by a back-end server*

146) pd_dbbuff_lock_release_detect = pipe | interval | switch

When the global buffer is accessed, a global buffer lock acquisition process occurs. This operand specifies the method that the process that is waiting for lock acquisition uses for detecting the lock release of the global buffer. This operand affects the processing performance when the access frequency to the same global buffer is high. The following table shows the processing mode for each operand specification and its characteristics.

Specification	HiRDB processing mode and characteristics	Characteristics	
		Response time	CPU usage
`pipe`	Lock-release processing uses a pipe file to inform the process waiting for lock acquisition of lock release. Response time is not affected by the degree of job multiplexing. For this reason, the response time is constant compared to specifying `interval` or `switch`.	Long	Low
`interval`	The process waiting for lock acquisition checks at regular intervals whether the lock is still in effect. Response time is affected by the degree of job multiplexing. For this reason, the greater the degree of job multiplexing, the longer the response time.	Short	High

Specification	HiRDB processing mode and characteristics	Characteristics	
		Response time	CPU usage
`switch`	The basic processing mode is the same as for `interval`. CPU usage is low compared to specifying `interval` because the CPU is used more efficiently. Tuning, however, is more difficult compared to when `interval` is specified.	Short	Medium

Note

Response time and CPU usage indicate general trends only. They might vary depending on the execution environment and the value specified for the `pd_dbbuff_lock_interval` or `pd_dbbuff_lock_spn_count` operand.

Specification guidelines

- If `switch` is specified when multiplexed UAP execution performance is notably slower than single unit performance, performance might improve.
- Whether `interval` or `switch` is better to specify (which one improves performance) depends on the OS, machine performance, type of UAP processing, multiplexed UAP executions count, and other factors. However, performance is generally more stable when `switch` is specified.

Relationship to other operands

When `interval` or `switch` is specified in this operand, check the values specified in the following operands:

- `pd_dbbuff_lock_spn_count`
- `pd_dbbuff_lock_interval`

147) pd_dbbuff_lock_spn_count = *number-of-spins-during-lock-acquisition-wait-processing*

~<unsigned integer>((0-2147483646))<<100>>

Specifies the number of spins during lock acquisition wait processing when `interval` or `switch` is specified in the `pd_dbbuff_lock_release_detect` operand.

An overview of the lock acquisition processing for the global buffer is provided below. This operand specifies how many times step 1 is repeated.

1. If the lock for the global buffer is released, the lock is acquired. If the lock is successfully acquired, the process is terminated. However, if the lock cannot be acquired, the process is repeated up to the number of times specified for this operand.
2. If acquisition fails in step 1, the system sleeps for the amount of time specified in the `pd_dbbuff_lock_interval` operand (waits for the specified amount of time to elapse).
3. Returns to step 1.

Condition

`interval` or `switch` must be specified in the `pd_dbbuff_lock_release_detect` operand.

Specification guidelines

Determine the value for this operand by referring to *Lock release contention wait generation rate in global buffer lock release processing* in the *HiRDB Version 9 System Operation Guide*.

Relationship to other operands

If this operand is specified, check the specification of the `pd_dbbuff_lock_interval` operand.

148) pd_dbbuff_lock_interval = *interval-during-lock-acquisition-wait-processing*

~<unsigned integer>((0-2147483647))<<1>>(milliseconds)

Specifies the interval during lock acquisition wait processing when `interval` or `switch` is specified in the `pd_dbbuff_lock_release_detect` operand.

An overview of the lock acquisition processing for the global buffer is provided below. For this operand, specify the time in step 2.

1. If the lock for the global buffer is released, the lock is acquired. If the lock is successfully acquired, the process is terminated. However, if the lock cannot be acquired, the process is repeated up to the number of times specified for the `pd_dbbuff_lock_spn_count` operand.
2. If acquisition fails in step 1, the system sleeps for the amount of time specified in this operand (waits for the specified amount of time to elapse).
3. Returns to step 1.

Condition

`interval` or `switch` must be specified in the `pd_dbbuff_lock_release_detect` operand.

Specification guidelines

Determine the value for this operand by referring to *Lock release contention wait generation rate in global buffer lock release processing* in the *HiRDB Version 9 System Operation Guide*.

149) pd_dbbuff_wait_interval = *global-buffer-occupation-state-check-interval*

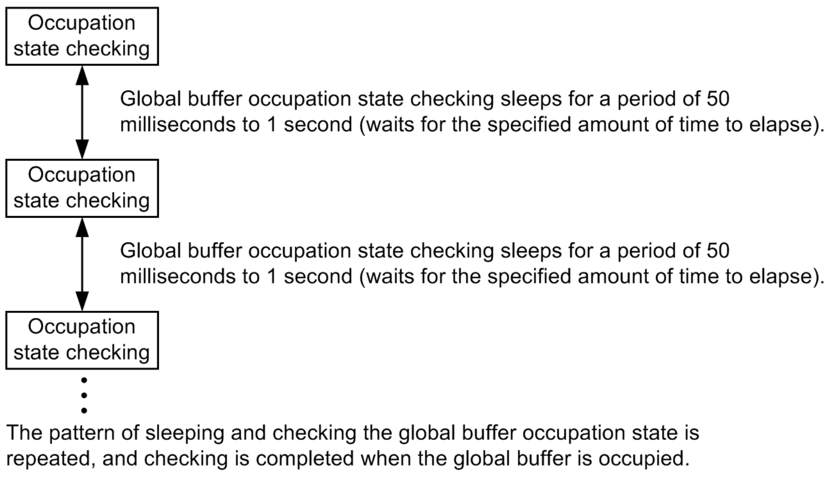
~<unsigned integer>((0-2147483647))(milliseconds)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the interval at which the global buffer occupation state is to be checked. Specifying this operand changes the method of checking the global buffer occupation state.

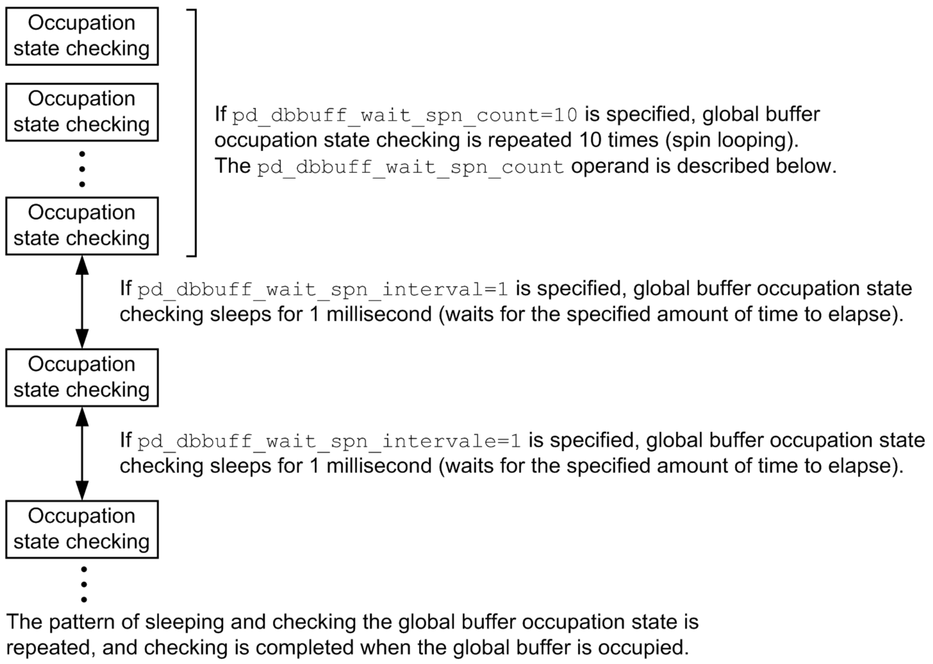
When this operand is not specified

The following processing occurs:


Occupation state checking
Global buffer occupation state checking sleeps for a period of 50 milliseconds to 1 second (waits for the specified amount of time to elapse).
Occupation state checking
Global buffer occupation state checking sleeps for a period of 50 milliseconds to 1 second (waits for the specified amount of time to elapse).
Occupation state checking
The pattern of sleeping and checking the global buffer occupation state is repeated, and checking is completed when the global buffer is occupied.

When this operand is specified

The following processing occurs:


Occupation state checking
Occupation state checking
Occupation state checking
If pd_dbbuff_wait_spn_count=10 is specified, global buffer occupation state checking is repeated 10 times (spin looping).
The pd_dbbuff_wait_spn_count operand is described below.
If pd_dbbuff_wait_spn_interval=1 is specified, global buffer occupation state checking sleeps for 1 millisecond (waits for the specified amount of time to elapse).
Occupation state checking
If pd_dbbuff_wait_spn_intervale=1 is specified, global buffer occupation state checking sleeps for 1 millisecond (waits for the specified amount of time to elapse).
Occupation state checking
The pattern of sleeping and checking the global buffer occupation state is repeated, and checking is completed when the global buffer is occupied.

Specification guidelines

Specify this operand when all of the conditions listed below are satisfied. Performance might improve. Typically when this operand is used, a value of `1` is specified.

- Global buffer lock-release wait has occurred. (You can check for this based on `WAITL` in the execution result of the `pdbufls` command.)
- You want to improve performance, even if doing so increases the CPU usage rate.

If the CPU usage has become too high because `1` was specified in this operand, increase the value. If there is unused capacity in the CPU usage rate when 1 is specified in this operand, increase the `pd_dbbuff_wait_spn_count` operand value. Performance might improve.

150) pd_dbbuff_wait_spn_count = *maximum-spin-loop-count-for-global-buffer-occupation-state-checking*

~<unsigned integer>((0-2147483646))<<0>>

This operand is applicable only to HiRDB/Parallel Server.

Specifies the maximum spin loop count in an interval loop that can occur during global buffer occupation state checking. For details, see the description of the `pd_dbbuff_wait_interval` operand.

Specification guidelines

Normally, there is no need to specify this operand. Specify this operand when you specify 1 in the `pd_dbbuff_wait_interval` operand.

151) pd_dbbuff_rate_updpage = *deferred-write-trigger-request-rate*

~<unsigned integer>((1-100))(%)

Specifies the trigger for deferred write, as a percentage of the number of buffers that have been updated.

Specification guidelines

Normally omit this operand. Deferred write processing might not always be completed within the synchronization point dump acquisition interval. You can specify this operand for such cases if you want to shorten the writing time by reducing the number of updated buffers and slightly reducing the updated buffer hit rate. A guideline for this operand's value is to use 50% (the initial value set by HiRDB) or determine the operand's value by referring to *Tuning deferred write processing* and *Tuning the synchronization point processing time when deferred write processing is used* in the *HiRDB Version 9 System Operation Guide*.

- When this operand is specified, the trigger remains constant, so the update buffer hit rate might decline as update volume increases, thus delaying response.

Relationships to other operands

This operand has the following relationships with the `pdbuffer` operand's `-y` option:

- The value set for the `pd_dbbuff_rate_updpage` operand applies to all the global buffers.

- The value of the `pdbuffer` operand's `-y` option applies to each global buffer.
- The `pdbuffer` operand's `-y` option takes precedence over the `pd_dbbuff_rate_updpage` operand.
- If the `pdbuffer` operand's `-y` option is omitted, the number of update buffer sectors for deferred write trigger event depends on the specification of this operand, as shown below:

pd_dbbuff_rate_updpage operand value	Number of update buffer sectors for deferred write trigger event
Specified	Number of global buffer sectors × `pd_dbbuff_rate_updpage` operand value
Omitted	Determined automatically by HiRDB

152) pd_dbbuff_trace_level = *global-buffer-control-information-trace-acquisition-level*

~<unsigned integer>((0-2147483647))<<0>>

Specifies the acquisition level for global buffer control information traces, as an unsigned integer. The acquisition level is calculated by totaling the level values that correspond to the different functions of the facility for acquiring global buffer control information traces. When `0` is specified, global buffer control information traces are not acquired. The facility for acquiring global buffer control information traces and its corresponding level value are shown below. For details about the facility for acquiring global buffer control information traces, see the *HiRDB Version 9 System Operation Guide*.

Facility for acquiring global buffer control information traces	Level value
Facility for acquiring `syncpoint` output synchronization control information	`1`

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reason such as performance checking, follow the maintenance engineer's instructions.

Effects on individual estimation formulas

If the value of the `pd_dbbuff_trace_level` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 5* under *Formulas for shared memory used by a single server*
- *Formula 4* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 5* under *Formulas for the size of the shared memory used by a back-end server*

2.2.27 Operands related to in-memory data processing

For details about in-memory data processing, see the manual *HiRDB Version 9 Batch Job Accelerator*.

153) pd_max_resident_rdarea_no = *maximum-number-of-in-memory-RDAREAs*

~<unsigned integer>((0-8388592))<<0>>

Specifies the maximum number of RDAREAs that can be made resident in memory (in-memory RDAREAs). RDAREAs in excess of the value specified in this operand cannot be made resident in memory.

Condition

A HiRDB Accelerator is needed in order to specify this operand.

Specification guidelines

- To perform in-memory data processing, you must specify at least `1` in this operand. If you specify `0`, in-memory data processing cannot be performed.
- For a HiRDB/Parallel Server, the value specified in this operand becomes the maximum for each back-end server.

Note

- Do not specify in this operand a larger value than necessary, because doing so increases the shared memory that HiRDB uses and might lead to shortages of shared memory, perhaps even preventing HiRDB from starting.
- The following notes apply when a value of `1` or a greater is specified for this operand in a Windows system that supports page fixing:

  ● If the shared memory used by the in-memory data buffer is fixed in the real memory (`pdmemdb -k stay -p fixed` is specified), the `pdntenv` command must be used to allocate the shared memory to the paging file. For details about the `pdntenv` command, see the manual *HiRDB Version 9 Command Reference*.

  ● If the shared memory used by the in-memory data buffer is fixed in the real memory (`pdmemdb -k stay -p fixed` is specified), when you estimate the required memory size, note that the shared memory is acquired with its size rounded up to the page size of a Windows large page. To check the page size of a Windows large page, use the `pdntenv -os` command. For the formula used to estimate the shared memory required for in-memory data processing, see *Storage Requirements for HiRDB* in the *HiRDB Version 9 Installation and Design Guide*.

  ● In the `SHMMAX` operand, specify a value that is rounded up to the page size of a Windows large page.

Effects on individual estimation formulas

If the value of the `pd_max_resident_rdarea_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 7* under *Formulas for shared memory used by a single server*
- *Formula 7* under *Formulas for the size of the shared memory used by a back-end server*

154) pd_max_resident_rdarea_shm_no = *maximum-number-of-shared-memory-segments-used-by-in-memory-data-buffer*

~<unsigned integer>((1-2147450879))<<pd_max_resident_rdarea_no value × 1.5>>

Specifies the maximum number of shared memory segments to be used by the in-memory data buffer. Shared memory segments in excess of the value specified in this operand cannot be used.

Condition

- A HiRDB Accelerator is needed in order to specify this operand.
- The `pd_max_resident_rdarea_no` operand must be specified.

Specification guidelines

- Consider specifying this operand if you will be performing in-memory data processing. See *Number of shared memory segments used by in-memory data buffers* in the *HiRDB Version 9 Installation and Design Guide* to determine an appropriate value to be specified in this operand.
- For a HiRDB/Parallel Server, the value specified in this operand becomes the maximum for each back-end server.

Notes

- Do not specify in this operand a larger value than necessary, because doing so increases the shared memory that HiRDB uses and might lead to shortages of shared memory, perhaps even preventing HiRDB from starting.
- The value specified in this operand is a maximum that HiRDB manages. It differs from the maximum value that each OS manages. For example, if `200` is specified in this operand but the maximum for your OS is 100, the maximum number of shared memory segments that can be secured will be 100.

Effects on individual estimation formulas

If the value of the `pd_max_resident_rdarea_shm_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 7* under *Formulas for shared memory used by a single server*
- *Formula 7* under *Formulas for the size of the shared memory used by a back-end server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

2.2.28 Operands related to table or index reservation count

155) pd_assurance_table_no = *table-reservation-count*

~<unsigned integer>((0-4194294500))<<500>>

Specifies the maximum number of tables that can use the *free space reusage facility*. For details about the free space reusage facility, see the *HiRDB Version 9 Installation and Design Guide*. For this operand, make sure that you enter the value described in *Specification guidelines*.

The free space reusage facility can be used if the number of target tables is no greater than the value of this operand. If the number of target tables is greater than this operand's value, the free space reusage facility can no longer be used.

Specification guidelines

The following explains how to determine this operand's value.

- Count each table as one.
- For a row-partitioned table, count each partition as one.
- For a HiRDB/Parallel Server, the specified value is applied to each back-end server. Obtain the value for each back-end server, and then specify the largest value in this operand.
- When the `pd_sysdef_default_option` operand is omitted or when `recommendable` or `v6compatible` is specified in the `pd_sysdef_default_option` operand

  Current number of tables using free space reusage facility + number of tables using free space reusage facility to be defined between next HiRDB startup and termination
- When `v7compatible` is specified in the `pd_sysdef_default_option` operand

  Current number of tables using free space reusage facility + MAX(*number of tables using free space reusage facility to be defined - between next HiRDB startup and termination*, 100)

Note

If the specified value is unnecessarily large, a shared memory shortage might prevent HiRDB (the back-end server for a HiRDB/Parallel Server) from starting. For the formula for calculating the size of shared memory, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand's default value depends on the value specified in the `pd_sysdef_default_option` operand, as shown below:

- `recommendable` (default value): `500`
- `v6compatible`: `0`
- `v7compatible`: `100`

Effects on individual estimation formulas

If the value of the `pd_assurance_table_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 2* under *Formulas for shared memory used by a single server*
- *Formula 2* under *Formulas for the size of the shared memory used by a back-end server*

156) pd_assurance_index_no = ***index-reservation-count***

~<unsigned integer>((50-4194294500))<<500>>

Specifies the number of indexes that can be used. For this operand, make sure that you enter the value described in *Specification guidelines*. If the number of indexes is equal to or less than the value of this operand, the index management information can be made resident in memory; this results in the following benefits:

- Performance is improved because index information is made resident in memory
- Statistical information is collected for indexes
- Output of duplicated messages is suppressed
- After free index pages have been released, the unused pages in used segments can be allocated preferentially.

If the number of indexes exceeds this operand value, these benefits are no longer applicable.

Specification guidelines

The following explains how to determine this operand value.

- Count each index as one.
- For a row-partitioned index, count each partition as one. For example, three partitions result in 3.
- For a HiRDB/Parallel Server, the specified value is applied to each back-end server. Obtain the value for each back-end server, and then specify the largest value in this operand.
- For a HiRDB/Single Server, if `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the index reservation count (240) for the data dictionary table must also be included. Add 240 to the obtained value.
- When the `pd_sysdef_default_option` operand is omitted or when `recommendable` is specified in the `pd_sysdef_default_option` operand

  Current number of indexes + number of indexes to be added between next HiRDB startup and termination
- When `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand

  MAX((*current number of indexes + number of indexes to be added between next HiRDB startup and termination*), (*current number of indexes* × 1.2))

Notes

If the specified value is unnecessarily large, a shared memory shortage might prevent HiRDB (the back-end server for a HiRDB/Parallel Server) from starting.

For the formulas used to calculate the size of shared memory, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand's default value depends on the value specified in the `pd_sysdef_default_option` operand, as shown below:

HiRDB server type	pd_sysdef_default_option operand value	Default value
HiRDB/Single Server	`recommendable` (default value)	500
	`v6compatible` or `v7compatible`	Number of indexes in data dictionary table (240) + 50
HiRDB/Parallel Server	`recommendable` (default value)	500
	`v6compatible` or `v7compatible`	50

2.2.29 Operands related to referential and check constraints

157) pd_constraint_name = **<u>LEADING</u>** | **TRAILING**

Specifies whether to specify a constraint name definition before or after constraint definition in referential constraint or check constraint.

`LEADING`:

Specifies a constraint name definition before constraint definition (standard SQL specification).

`TRAILING`:

Specifies a constraint name definition after constraint definition (XDM/RD-compatible specification).

158) pd_check_pending = <u>USE</u> | NOUSE

Specifies whether the check pending status (state that prevents the use of data for which integrity can no longer be guaranteed) is to be used in referential constraint or check constraint.

`USE`: Use the check pending state.

`NOUSE`: Do not use the check pending state.

Specification guidelines

Normally specify `USE`. If `NOUSE` is specified in this operand, data integrity might not be guaranteed. If processing performance is more important then data integrity, or if neither referential constraint nor check constraint are used, specify `NOUSE`.

Notes

If `USE` is specified in this operand or the operand is omitted, HiRDB does the following when it executes the `PURGE TABLE` statement:

- Temporarily locks the data dictionary table (resource type: `3005`, type name: `DICT`) by placing it in EX mode.
- Locks the data dictionary RDAREA (resource type: `0001`, type name: `RDAR`) by placing it in SU mode until the transaction is terminated.

Therefore, any other command or a utility that attempts to lock these resources might not be concurrently executable. If a command or a utility satisfies the conditions listed below, do not attempt to execute the command or utility concurrently.

Command or utility	Condition
`pdmod` (database structure modification utility)	The attribute definition of a data dictionary is changed.
`pdcopy` (database copy utility)	The following two conditions are satisfied: • `x` or `r` is specified in the `-M` option. • The data to be copied includes a master directory RDAREA or a data dictionary RDAREA.
`pdreginit` (registry facility initialization utility)	`all` is specified in the `-k` option.

2.2.30 Operands related to temporary tables

159) pd_max_tmp_table_rdarea_no = *maximum-number-of-temporary-table-RDAREAs*
~<unsigned integer> ((0-131088))<<0>>

Specifies the maximum number of temporary table RDAREAs. If the total number of temporary table RDAREAs exceeds this operand value, HiRDB cannot start.

Specification guidelines

- Specify the total number of temporary table RDAREAs including those that will be added in the future.
- For a HiRDB/Parallel Server, this operand value is applied for each back-end server. Therefore, as a guideline, specify this operand value based on the back-end server that uses the largest number of temporary table RDAREAs.

Notes

- This operand value affects the size of shared memory used by HiRDB; therefore, do not specify a value that is greater than the value described in *Specification guidelines*. If the specified value is greater than the value described in *Specification guidelines*, HiRDB might not be able to start due to a shortage of shared memory.
- Make sure that this operand's value is less than the `pd_max_rdarea_no` operand's value.

- If the `pd_max_temporary_object_no` operand's value is `1` or greater, this operand must also specify a value of `1` or a greater.

Relationship to other operands

This operand is related to the following operands:

- `pd_max_rdarea_no`
- `pd_max_temporary_object_no`

Effects on individual estimation formulas

If the value of the `pd_max_tmp_table_rdarea_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 9* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

160) pd_max_temporary_object_no = *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*

~<unsigned integer> ((0-131072))<<0>>

Specifies the maximum number of temporary tables and temporary table indexes that can be used at any one time for each server.

Specification guidelines

Use the formula shown below to determine the value of this operand. For a HiRDB/Parallel Server, obtain the value for each back-end server.

Maximum value of (*the number of transaction-specific temporary tables*
+ *total number of temporary table indexes for those temporary tables*) × *number of activities*#
+ (*number of SQL session-specific temporary tables used in the SQL session*
+ *total number of temporary table indexes for those temporary tables*) × *number of connected users*

#

number of activities:

((*value of pd_max_users* + 3) × 2 + 1) + α

α : If the value specified for `pd_max_users` is `60` or less, 5; if it is `61` or greater, 0.

For a HiRDB/Parallel Server, this operand's value is applied to each back-end server. Therefore, as a guideline, specify the largest value used among all back-end servers in this operand.

Notes

The value of this operand affects the size of shared memory used by HiRDB; therefore, do not specify a value that is greater than the value described in *Specification guidelines*. If the specified value is greater than the value described in *Specification guidelines*, HiRDB might not be able to start due to a shortage of shared memory.

Relationship to other operands

This operand is related to the `pd_max_tmp_table_rdarea_no` operand.

Effects on individual estimation formulas

If the value of the `pd_max_temporary_object_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 8* under *Formulas for shared memory used by a single server*
- *Formula 8* under *Formulas for the size of the shared memory used by a back-end server*

161) pd_tmp_table_initialize_timing = <u>STARTING</u> | ACCESS

Temporary table RDAREAs are normally initialized when HiRDB starts. Specify this operand if you want to change when the temporary table RDAREAs are initialized to a time other than when HiRDB starts. For details about the initialization of temporary table RDAREAs, see the *HiRDB Version 9 Installation and Design Guide*.

`STARTING`:

When HiRDB starts, initialize all temporary table RDAREAs that were used during the previous session.

`ACCESS`:

When the first `INSERT` statement is executed on a temporary table, initialize the temporary table RDAREA that has become the storage candidate. For details about the temporary table RDAREA that becomes the storage candidate, see *Rules for choosing an RDAREA for storage* in the *HiRDB Version 9 Installation and Design Guide*.

Notes

If `ACCESS` is specified, the overhead of initializing the temporary table RDAREA occurs when the first `INSERT` statement is executed on a temporary table since HiRDB started. Therefore, the transaction might be canceled for the following reasons:

- The lock-release wait time was exceeded.
- The HiRDB client's maximum wait time was exceeded.

You can determine the amount of overhead involved in initializing the temporary table RDAREA (number of directory pages that are initialized) using the following formula:

$$\uparrow 1{,}000 \div (50 \times (\textit{page size} \div 2{,}048)) \uparrow$$
$$+ \sum \uparrow \textit{segment size of each HiRDB file}$$
$$\div (16{,}000 \times (\textit{page size} \div 2{,}048)) \uparrow$$
(pages)

Take into account the above overhead time when you specify the operands shown in *Relationship to other operands*.

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_wait_timeout`
- `PDCWAITTIME` in the client environment definition

2.2.31 Operands related to HiRDB file system areas

162) pd_large_file_use = Y | <u>N</u>

Specifies whether to use a HiRDB file system area with a size of 2,048 MB or greater.

`Y`:

Use a HiRDB file system area with a size of 2,048 MB or greater. When `Y` is specified, the maximum size of a HiRDB file system area is set to 1,048,575 MB. If you will be using the system log file automatic extension facility, specify `Y`.

`N`:

Do not use a HiRDB file system area with a size of 2,048 MB or greater.

Notes

- If you specify `N` in this operand and create a HiRDB file system area with a size of 2,048 megabytes or greater, it will not be possible to open the HiRDB files stored in the HiRDB file system area.
- Be careful when changing the specification of this operand from `Y` to `N`. If this change is made, RDAREAs and system files in HiRDB file system areas with a size of 2,048 megabytes or greater are placed in error shutdown status. To free these RDAREAs from error shutdown, use the following procedure.

To free RDAREAs from error shutdown:

1. Terminate HiRDB normally with the `pdstop` command.
2. Change the value of the `pd_large_file_use` operand to `Y`.
3. Start HiRDB normally with the `pdstart` command.
4. Use the `pdrorg` command to unload data from the RDAREAs in the HiRDB file system area with a size of 2,048 MB or greater.

5. Terminate HiRDB normally with the `pdstop` command.
6. Change the value of the `pd_large_file_use` operand to `N`.
7. Start HiRDB normally with the `pdstart` command.
8. Use the `pdmod` command to reinitialize the RDAREAs in the HiRDB file system area with a size of 2,048 MB or greater. Allocate a HiRDB file system area with a size of less than 2,048 MB to the RDAREAs.
9. Use the `pdrorg` command to reload the data that was unloaded in step 4.

If a system file goes into error shutdown, take corrective action as directed in the message that is issued.

Relationship to other operands

When `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default for this operand is `N`.

Effects on individual estimation formulas

If the value of the `pd_large_file_use` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

163) pd_ntfs_cache_disable = <u>Y</u> | N

Specifies whether to use the no-cache access method to access HiRDB file system areas. This operand takes effect on a non-NTFS file system such as FAT.

`Y`:

Use the no-cache access method. Because Windows' cache memory is not used, Windows cache speed has no effect on processing, thereby achieving stable HiRDB performance. Additionally, the amount of Windows resources required is less.

When specifying `Y` in this operand, tune the global buffer; otherwise, performance decreases.

`N`:

Do not use the no-cache access method. In this case, Windows' cache memory is used.

Specification guidelines

Specify `Y` in this operand in the following cases:

- Message `KFPO00107-E` or `KFPS00700-E` is issued.
- DLL initialization error occurs.
- The combined total size* of the files to be concurrently opened by a single server machine exceeds 100 GB.

  * Use the combined total size for the following files as a guideline:
  - HiRDB file system area for RDAREA
  - HiRDB file system area for system files
  - Other HiRDB file system areas that are open
  - User-specified files that are open
  - Files that have been opened by other programs

Notes

The no-cache access method might not be available, depending on the purpose of the HiRDB file system area. The following table shows the support of the no-cache access method:

Purpose of HiRDB file system area	Support
HiRDB file system area for RDAREAs (DB)	Y
HiRDB file system area for RDAREA (SDB)	--#1
HiRDB file system area for system files (SYS)	Y

Purpose of HiRDB file system area	Support
HiRDB file system area that does not use the Windows cache, for utilities (NUTL)	Y#2
HiRDB file system area for work table files or list RDAREAs (WORK)	--
HiRDB file system area for utilities (UTL)	--
HiRDB file system area with no specific purpose defined (SVR)	--

Y: The no-cache access method is available.

--: The no-cache access method is not available.

#1: Cannot be created in a Window file system. This is used exclusively for direct disk access.

#2: The no-cache access method is used regardless of the value specified for the `pd_ntfs_cache_disable` operand.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `N`.

2.2.32 Operands related to the facility for predicting reorganization time

For details about how to use the facility for predicting reorganization time, see the *HiRDB Version 9 System Operation Guide* and *HiRDB Version 9 Command Reference*.

164) pd_rorg_predict = Y | N

Specifies whether to use the facility for predicting reorganization time.

`Y`: Uses the facility for predicting reorganization time.

`N`: Does not use the facility for predicting reorganization time.

Notes

- To execute the facility for predicting reorganization time, you must create a data dictionary RDAREA for storing the analysis information table and the operation history table.
- If you specify `Y` for this operand, shared memory is used. For details about the size of the shared memory to be used, see the *HiRDB Version 9 Installation and Design Guide*.

2.2.33 Operands related to security

(1) Operands related to the security audit facility

For details about how to use the security audit facility, see the *HiRDB Version 9 System Operation Guide*.

165) pd_audit = Y | N

Specifies whether to begin collecting an audit trail when HiRDB (a unit for a HiRDB/Parallel Server) is started.

`Y`: Begins collecting an audit trail when HiRDB is started.

`N`: Does not begin collecting an audit trail when HiRDB is started.

Even if `Y` is specified for this operand, you can still collect an audit trail by executing the `pdaudbegin` command.

Conditions

All of the following conditions must be satisfied. If `Y` is specified when all of these conditions are not satisfied, HiRDB (a unit for a HiRDB/Parallel Server) cannot be started.

- A HiRDB file system area has been created for an audit trail file.
- The name of the HiRDB file system for the audit trail file is specified for the `pd_aud_file_name` operand.

166) pd_aud_file_name = *HiRDB-file-system-area-name-for-audit-trail-file*

~<path name>((maximum of 150 characters))

This operand is required if you use the security audit facility. If you do not specify this operand, you cannot use the security audit facility.

Specify an absolute path name for the name of the HiRDB file system area for an audit trail file.

When you use the security audit facility on a HiRDB/Parallel Server, We recommend that you acquire an audit trail for the entire system. To do so, specify one of the following.

- `pd_aud_file_name` operand in the system common definition
- `pd_aud_file_name` operands in all of the unit control information definitions

However, in system configurations that run multiple units on a single server machine, the `pd_aud_file_name` operands must be specified in all unit control information definitions.

Notes

- When this operand is specified, HiRDB (a unit for a HiRDB/Parallel Server) cannot be started if an error occurs during the access to the HiRDB file system area for audit trail files.
- If the same audit trail file is specified in the `pd_aud_file_name` operands in the system common definition for multiple units on the same server machine, the correct audit trail cannot be acquired.

167) pd_aud_max_generation_size = *audit-trail-file-maximum-size*

~<unsigned integer>((1-5240))<<100>> (megabytes)

Specifies, in megabytes, the maximum size of audit trail files.

Specification guidelines

- Because HiRDB needs 20 MB for management, determine the value for this operand so that the following condition is satisfied:

  pd_aud_max_generation_size-value × *pd_aud_max_generation_num-value* < *size-of-HiRDB-file-system-area-for-audit-trail-files* (value of the `-n` option of the `pdfmkfs` command) - 20 MB

- When the specified value is smaller than the capacity of one audit trail record or no value is specified for this operand, and the size of one audit trail record is larger than the default, it will not be possible for HiRDB to start.

  To start such a HiRDB unit, set the operand value so that the following condition is satisfied:

  value of pd_aud_max_generation_size ≥ ↑ *maximum-audit-trail-record-size* ÷ 1,024 ↑ × 1,024 + 2,048 (bytes)

  Use the following formula to calculate the audit trail record size:

  maximum-audit-trail-record-size = 1,067 + ↑ *value of pd_aud_sql_source_size* ÷ 4 ↑ × 4 + ↑ *value of pd_aud_sql_data_size* ÷ 4 ↑ × 4 (bytes)

- If data is being output to the audit trails asynchronously, see *Output to audit trail file (asynchronous output)* in the *HiRDB Version 9 System Operation Guide*.

168) pd_aud_max_generation_num = *maximum-audit-trail-file-count*

~<unsigned integer>((2-200))<<50>>

Specifies the maximum number of (number of generations of) audit trail files to be created inside the HiRDB file system area for audit trail files.

Specification guidelines

- We recommend that you not specify the maximum value (200) in case errors occur in all audit trail files. For details about how to handle errors in audit trail files, see the *HiRDB Version 9 System Operation Guide*.
- Because HiRDB needs 20 MB for management, determine the value for this operand so that the following condition is satisfied:

pd_aud_max_generation_size-value × *pd_aud_max_generation_num-value* < *size-of-HiRDB-file-system-area-for-audit-trail-files* (value of the `-n` option of the `pdfmkfs` command) - 20 MB

Notes

During startup of HiRDB (a unit for a HiRDB/Parallel Server), if there is a file with a generation number that is greater than the value specified for this operand, the specified value becomes invalid. In this case, the largest generation number is assumed as the maximum number of audit trail files to be created inside the HiRDB file system area.

169) pd_aud_no_standby_file_opr = down | <u>forcewrite</u>

Specifies the processing to be performed by HiRDB when no swappable audit trail file is available or when all sectors of the asynchronous output buffer are awaiting a flush.

`down`:

If the number of remaining swappable audit trail files reaches one, HiRDB (or a unit for a HiRDB/Parallel Server) is forcibly terminated. If the audit trail file becomes full or an error occurs in the current file, the `down` setting takes effect and swapping occurs. For details about how to proceed in the event HiRDB is terminated forcibly due to this operand specification, see *When HiRDB is terminated forcibly because there are no swappable target* in the *HiRDB Version 9 System Operation Guide*.

When all sectors of the asynchronous output buffer are awaiting a flush, HiRDB (or, for a HiRDB/Parallel Server, the unit) is terminated forcibly. For the corrective action to take thereafter, see *When all sectors of the asynchronous output buffer are placed in flush-wait status* in the *HiRDB Version 9 System Operation Guide*.

`forcewrite`:

If no swappable audit trail file is available, an audit trail file waiting for data loading (except for files that are shut down) is forcibly made into a swapping target, and audit trail outputting is continued. For this process, the audit trail file waiting for data loading that has the oldest update date is made into the swapping target.

If the `pdaudswap` command is executed to swap files or all files are shut down, the output of audit trails is stopped.

When all sectors of the asynchronous output buffer are awaiting a flush, audit trail output resumes by forcibly overwriting the asynchronous output buffer that initially began awaiting a flush and reusing it. The audit trail in the asynchronous output buffer that was overwritten is destroyed.

170) pd_aud_async_buff_size = *size-of-buffer-used-for-asynchronous-output-of-audit-trail-file*

~<unsigned integer>((0, 4096-6553600))<<401408>> (bytes)

Specifies the size (bytes) of the buffer to be used for asynchronously outputting the audit trail. If `0` is specified, the audit trail is synchronously output. When the specified value is smaller than the maximum audit trail record size or no value is specified for this operand, and the maximum audit trail record size is larger than the default, it will not be possible for the HiRDB unit to start.

For details about the maximum audit trail record size, see the description of the `pd_aud_max_generation_size` operand.

The following table describes the advantages and disadvantages of each output method.

pd_aud_async_buff_size value	Audit trail output method	Advantages	Disadvantages
`0`	Synchronous output	Audit trail can be reliably output to an audit trail file.	Because file input/out occurs on the extension of SQL processing, the impact on performance is large.
`4096-6553600`	Asynchronous output	Can reduce the impact on SQL processing performance.	If HiRDB (a unit for a HiRDB/ Parallel Server) is abnormally terminated after the audit trail is output to the buffer and before it is output to an audit trail file, the audit trail might be lost.

Specification guidelines

To output an audit trail asynchronously, we recommend that you set this buffer size on the large side. There is only one of these buffers per unit, so performance might be degraded if contention occurs among environments that have many transactions that create high processing loads.

Operand rule

For this operand, specify an integer multiple of 4,096. If a value that is not an integer multiple of 4,096 is specified, it is rounded up to an integer multiple of 4,096 and set as the value for this operand. For example, if `5000` is specified, `8192` is set for the operand.

Notes

- Starting HiRDB (or, for a HiRDB/Parallel Server, the unit) requires shared memory for the unit controllers equal in size to *value of pd_aud_async_buff_size* × *value of pd_aud_async_buff_count* (bytes). Make sure that the value from this equation does not exceed the upper limit for shared memory for unit controllers as a whole. For details about calculating the shared memory size used by unit controllers, see the *HiRDB Version 9 Installation and Design Guide*.

- When the values specified in the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands are small, all sectors of the asynchronous output buffer might wait for flushing, transaction execution times might lengthen, or, depending on the specification of the `pd_aud_no_standby_file_opr` operand, HiRDB (or, for a HiRDB/Parallel Server, the unit) might be forcibly terminated.

  Determine the settings for the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands taking into consideration the number of audit trail outputs per unit of time. For details, see *Output to audit trail file (asynchronous output)* in the *HiRDB Version 9 System Operation Guide*.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `4096`.

Effects on individual estimation formulas

If the value of the `pd_aud_async_buff_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

171) pd_aud_async_buff_count = *number-of-buffer-sectors-used-for-asynchronous-output-of-audit-trail-file*

~<unsigned integer>((1-6500))<<max(1, number of HiRDB servers in unit × 10)>>

Specifies the number of buffer sectors to be used for asynchronously outputting an audit trail.

Specification guidelines

We recommend that the number of buffer sectors be set on the high side. If the value is too small, writing to buffers might take longer due to writing into audit trail files, which can degrade performance.

Notes

- Starting HiRDB (or, for a HiRDB/Parallel Server, the unit) requires shared memory for the unit controllers equal in size to *value of pd_aud_async_buff_size × value of pd_aud_async_buff_count* (bytes). Make sure that the value from this equation does not exceed the upper limit for shared memory for unit controllers as a whole. For details about calculating the shared memory size used by unit controllers, see the *HiRDB Version 9 Installation and Design Guide*.
- When the values specified in the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands are small, all sectors of the asynchronous output buffer might wait for flushing, transaction execution times might lengthen, or, depending on the specification of the `pd_aud_no_standby_file_opr` operand, HiRDB (or, for a HiRDB/Parallel Server, the unit) might be forcibly terminated.

  Determine the settings for the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands taking into consideration the number of audit trail outputs per unit of time. For details, see *Output to audit trail file (asynchronous output)* in the *HiRDB Version 9 System Operation Guide*.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `3`.

Effects on individual estimation formulas

If the value of the `pd_aud_async_buff_count` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

172) pd_aud_async_buff_retry_intvl = ***retry-interval-for-allocation-of-a-buffer-to-be-used-for-asynchronous-output-of-audit-trail-file***

~<unsigned integer>((1-1000))<<50>> (milliseconds)

Specifies the retry interval for monitoring for a buffer to be used for asynchronous output of the audit trail so that the audit trail can be acquired when all buffers are in use.

Specification guidelines

Normally, there is no need to specify this operand.

When the security audit facility is used and a UAP requires an extended amount of time to execute, specifying a small value in this operand might reduce the UAP execution time.

173) pd_aud_sql_source_size = ***size-of-sql-statement-output-to-audit-trail***

~<unsigned integer>((0-2000000))<<0>>(bytes)

Specifies the size in bytes of the SQL statements output to the audit trail when using the security audit facility. When `0` is specified, no SQL statements are output to the audit trail. For SQL statements larger than the specified value, the portion in excess of the specified value is not output to the audit trail.

Relationship to other operands

If you specify this operand, re-estimate the specifications for the `pd_aud_max_generation_size` and `pd_aud_async_buff_size` operands.

Effects on individual estimation formulas

If the value of the `pd_aud_sql_source_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining audit trail file capacity*

174) pd_aud_sql_data_size = ***size-of-sql-data-output-to-audit-trail***

~<unsigned integer>((0-1000000))<<0>>(bytes)

Specifies the size in bytes of the SQL data output to the audit trail when using the security audit facility. When `0` is specified, no SQL data is output to the audit trail. For SQL data larger than the specified value, the portion in excess of the specified value is not output to the audit trail.

Relationship to other operands

If you specify this operand, re-estimate the specifications for the `pd_aud_max_generation_size` and `pd_aud_async_buff_size` operands.

Effects on individual estimation formulas

If the value of the `pd_aud_sql_data_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining audit trail file capacity*

175) pd_aud_file_wrn_pnt = ***warning-message-output-trigger*****[,*****trigger-for-resetting-warning-message-output-status*****]**

warning-message-output-trigger**:~<unsigned integer><<0-100>><<0 or 80>>(%)**

When the number of unswappable audit trail files reaches or exceeds the warning value, a warning message is issued. For this operand, specify the warning value as a percentage of the maximum audit trail file count specified in the `pd_aud_max_generation_num` operand. For example, if `100` is specified for the `pd_aud_max_generation_num` operand, and `90` is specified for the `pd_aud_file_wrn_pnt` operand, the `KFPS05123-W` warning message is issued when the number of unswappable audit trail files reaches or exceeds 90.

For a HiRDB/Parallel Server, the number is checked for each unit.

If `0` is specified in this operand, no warning message is issued.

Relationship to other operands

- If this operand is omitted, and if `MANUAL` is specified for the `pd_watch_resource` operand or this operand is omitted, `0` is assumed for the `pd_aud_file_wrn_pnt` operand. That is, no warning message is issued.

- If this operand is omitted, and if `AUTO` is specified for the `pd_watch_resource` operand or this operand is omitted, `80` is assumed for the `pd_aud_file_wrn_pnt` operand. That is, a warning message is issued when 80% is reached or exceeded.

***trigger-for-resetting-warning-message-output-status*:~<unsigned integer><<0-99>>(%)**

Specifies the trigger for resetting the warning message output status. When the warning message (`KFPS05123-W`) is output, HiRDB goes into the warning message output status. Once HiRDB goes into this status, the warning message is not output again, even if the number of unswappable audit trail files exceeds the warning value again. However, when the number of unswappable audit trail files falls below the trigger for resetting the warning message output status specified here, the warning message output status is reset.

For example, if `pd_aud_file_wrn_pnt=90,70` is specified, the warning message is output when the number of unswappable audit trail files reaches or exceeds 90% of the maximum number of audit trail files. Afterwards, no warning message is output until the number of unswappable audit trail files falls below 70% of the maximum number of audit trail files. After the percentage falls below 70%, and when it subsequently reaches or exceeds 90% again, the warning message is output.

Notes

- When this specification is omitted, warning-message-output-trigger `-30` is assumed as the default (if the result is a negative number, `0` is used).
- If a value greater than the warning message output trigger is specified, the warning message output trigger value is used.

176) pd_aud_auto_loading = Y | <u>N</u>

Specifies whether the facility for automatically loading audit trail table data is to be used. For details about the facility for automatically loading audit trail table data, see the *HiRDB Version 9 System Operation Guide*.

`Y`:

Uses the facility for automatically loading audit trail table data. When this value is specified, data is loaded into the audit trail table automatically using as the trigger generation swapping of audit trail files (excluding swaps caused by errors).

`N`:

Does not use the facility for automatically loading audit trail table data. Data will not be loaded automatically into the audit trail table; instead, it must be loaded manually by the auditor.

Specification guidelines

Use of the facility for automatically loading audit trail table data can reduce the auditor's workload. However, data loading begins during online transactions, so there is an increase in the number of I/Os to the CPU and disks, thus increasing the system load. Decide whether to use the facility for automatically loading audit trail table data based on these considerations.

Note

When this operand is set to `Y` in a HiRDB/Parallel Server, a HiRDB file system area for audit trail files must be created in each unit that has a system manager, and the `pd_aud_file_name` operand must be specified. If it is not specified in such a case, the facility for automatically loading audit trail table data cannot be used.

Effects on individual estimation formulas

If the value of the `pd_aud_auto_loading` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Processes started by HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

177) pdaudload

[-i *index-creation-method*]

[-l *log-acquisition-mode*]

[-n [*batch-output-local-buffer-sector-count*],,

[*random-access-local-buffer-sector-count*]]

[-y]

[-X *response-monitoring-time-for-server-to-server-communication*]

[-S *sort-buffer-size*]

Specifies environment information for the database creation utility (`pdload`) that is activated when the facility for automatically loading audit trail table data is used.

Condition

`Y` must be specified in the `pd_aud_auto_loading` operand.

Specification guidelines

For guidance in determining the operand values, see *Applicable conditions* under *Facility for automatically loading audit trail table data* in the *HiRDB Version 9 System Operation Guide*.

Notes

If this operand is specified more than once, only the values specified in the first instance of the operand are valid, and values specified in subsequent instances are ignored.

-i *index-creation-method*

~<<c>>

Specifies the index creation method:

`c`:

Batch creation mode. While row data is being stored, index creation information is output to an index information file without creating the index. When storage processing for all row data has been completed, the index is created.

`s`:

Index update mode. The index is updated each time a row of data is stored.

-l *log-acquisition-mode*

~<<p>>

Specifies the method of acquiring the database update log when `pdload` is executed:

`a`:

Log acquisition mode. Database update logs required for rollbacks and rollforwards are acquired.

`p`:

Pre-update log acquisition mode. Database update log required for rollbacks is acquired, but no database update log required for rollforwards is acquired.

-n [*batch-output-local-buffer-sector-count*],,[*random-access-local-buffer-sector-count*]

Specifies that local buffers are to be used to load data into a table. When this option is specified, the database can be accessed using local buffers, which reduces the number of I/Os because batch output is used.

When this option is omitted, output is in units of pages using global buffers.

When *batch-output-local-buffer-sector-count* is omitted and only *random-access-local-buffer-sector-count* is specified, enclose the specification, including the commas, in double quotation marks. For example, to omit *batch-output-local-buffer-sector-count* and specify `1000` as the *random-access-local-buffer-sector-count* of `1000`, specify as follows.

```
pdaudload -n ",,1000"
```

However, you cannot omit both *batch-output-local-buffer-sector-count* and *random-access-local-buffer-sector-count*. If both are omitted but the `-n` option is specified, a definition error will result and the `KFPS01895-E` message will be issued.

batch-output-local-buffer-sector-count: ~<unsigned integer>((2-4096))

Specifies the number of batch output local buffer sectors. The batch output local buffer is used for the database.

random-access-local-buffer-sector-count: ~<unsigned integer>((4-125000))

Specifies the number of random access local buffer sectors. The random access local buffer is used for index pages.

-y

Specifies that when all unused pages have been used during data loading, data must be stored in unused areas of pages that are being used. When this option is specified, the `KFPH26010-I` message is issued prior to storing data in unused areas on pages that are being used.

When you specify this option, specify `a` in the `-l` option. Specifying `p` in the `-l` option or not specifying the `-l` option will result in a definition error and issuance of the `KFPS01895-E` message.

-X *response-monitoring-time-for-server-to-server-communication*

~<unsigned integer>((1-65535))<<300>>(seconds)

Specifies a response monitoring time, in seconds, for dictionary operations. When execution time exceeds the time set in this option during a dictionary operation, `pdload` determines that an error has occurred in the access to the dictionary and halts processing with a return code of `8`. When processing is halted, automatic data load processing of the audit trail table is also stopped.

The purpose of this option is so that `pdload` can monitor the response time of communications for dictionary operations performed by commands in order to detect errors. This is important, because when an error occurs in communication with the server that executed a command, the command might become unresponsive or the transaction might stop.

-S *sort-buffer-size*

~<unsigned integer>((128-2097152))<<1024>>(kilobytes)

Specifies the buffer size, in kilobytes, of the work file for sorting that is used when data is loaded into the audit trail table defined for an index during index batch creation mode (with `-i c` specified). The buffer is secured in the single server in the case of a HiRDB/Single Server and in the audit trail table in the case of a HiRDB/Parallel Server.

For details about the size of the sort buffer, see *sort statement (specification of sort work directory information)* under *Database Load Utility (pdload)* in the manual *HiRDB Version 9 Command Reference*.

(2) Operands related to strengthening security

178) pd_security_host_group = "*host-name*"[, "*host-name*"]...

Specifies explicitly the hosts to be used in the HiRDB server configuration and limits HiRDB operations that might affect security (operations from hosts other than those defined by utilities and related program products). Use of this operand is applicable in constructing a system that requires a high level of security protection.

You must specify in this operand all hosts to be used on the network that constitutes the HiRDB server. Specifying this operand can reduce security risks. Specify a host by its IP address or in FQDN format; a loopback address can also be specified.

This operand's specification can be modified at a forced, abnormal, or planned termination.

A host name specification must not exceed 256 characters in length.

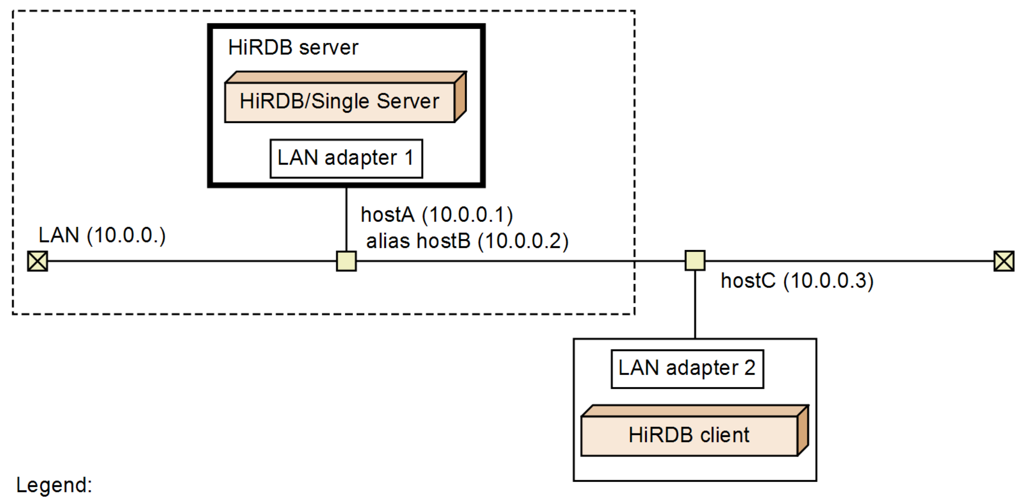
Notes

- When a DNS server is used, you must also register in the DNS server all hosts specified in this operand. If no DNS server is used, register the host names specified here in the `hosts` file.
- If a name cannot be resolved, the `KFPS04693-E` message is issued and HiRDB startup processing terminates.
- If the same host name, IP address, or FQDN character string is specified more than once, the `KFPS04693-E` message is issued and HiRDB startup processing terminates. No error results when the IP address is the same but the host names are different.
- When a loopback address is specified in the `-x` option of the `pdunit` operand, also specify a loopback address in this operand.

Comments

Examples of specifying the `pd_security_host_group` operand are shown below.

pd_security_host_group operand specification example (for a HiRDB/Single Server)

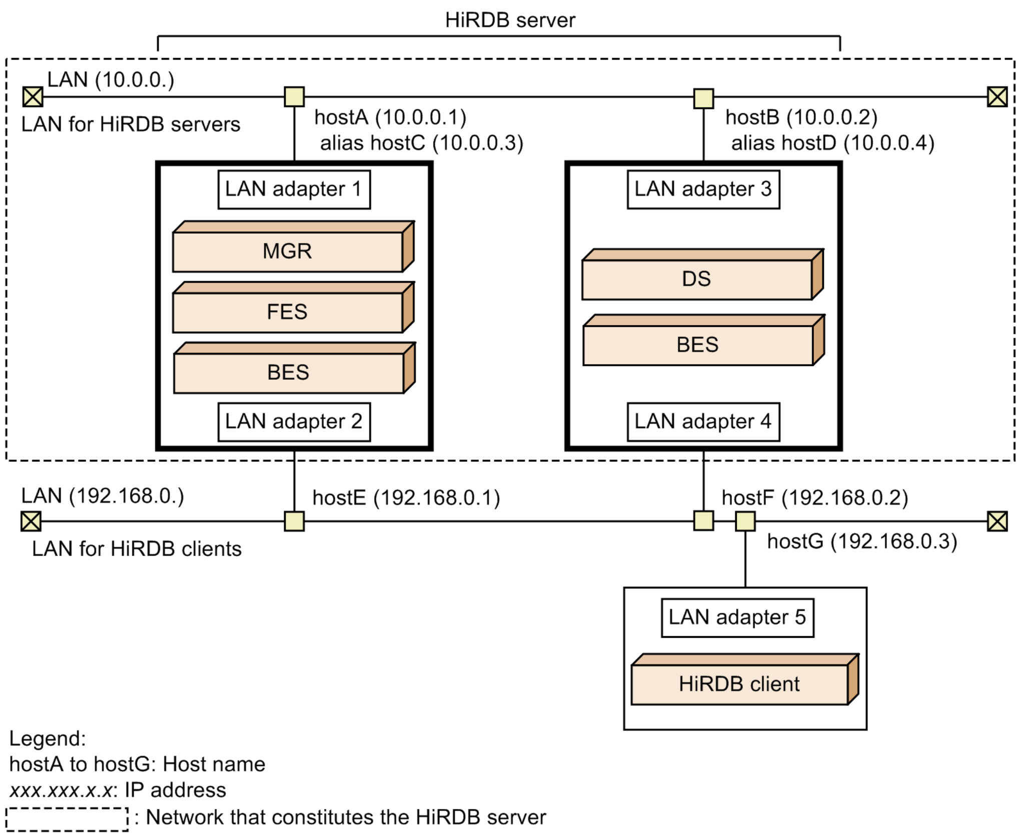


Legend:
hostA to hostC: Host name
xx.x.x.x : IP address
⬚ : Network that constitutes the HiRDB server

- System common definition coding

```
                         :
  set pd_security_host_group=hostA,hostB
  pdstart -t SDS -s sds -x hostA
                         :
```

pd_security_host_group operand specification example (for a HiRDB/Parallel Server)



- System common definition coding

```
                          :
set pd_security_host_group=hostA,hostB,hostC,hostD,hostE,hostF
pdstart -t FES -s fes1 -x hostA
                          :
```

2.2.34 Operands related to the system switchover facility

179) pd_ha = use | <u>nouse</u>

Specifies whether the system switchover facility is to be used.

`use`: Use the system switchover facility.

`nouse`: Do not use the system switchover facility.

180) pd_ha_ipaddr_inherit = <u>Y</u> | N

Specifies whether IP addresses are to be inherited when the system switchover facility is used. How the system switchover facility operates depends on whether IP addresses will be inherited; for details, see the *HiRDB Version 9 System Operation Guide*.

`Y`: Inherit IP addresses.

`N`: Do not inherit IP addresses.

In server mode operations, IP addresses cannot be inherited.

Omit this operand when you use the standby-less system switchover facility (this operand will be ignored if specified in this case).

Specification guidelines

- HiRDB/Single Server

  To inherit IP addresses, specify `Y`; otherwise, specify `N`. However, in server mode operations, IP addresses cannot be inherited. Therefore, if you specify `Y` here, you need to specify `N` for the `pd_ha_ipaddr_inherit` operand of the unit control information definition.

Note that if you specify `N` for this operand, you cannot specify `Y` for the `pd_ha_ipaddr_inherit` operand of the unit control information definition.

- HiRDB/Parallel Server

  To inherit IP addresses, specify `Y`; otherwise, specify `N`. Note that the units that are the targets of the server mode cannot inherit IP addresses. Therefore, if you specify `Y` here, you need to specify `N` for the `pd_ha_ipaddr_inherit` operand of the unit control information definition of the units that are the targets of the server mode.

  If you specify `N` for this operand, you cannot specify `Y` for the `pd_ha_ipaddr_inherit` operand of the unit control information definition.

181) pd_ha_switch_timeout = Y | N

This operand can be specified when the server mode is used. Specification of this operand is invalid in the monitor mode.

This operand specifies whether to switch the system without waiting for internal termination processing of HiRDB when internal termination processing of HiRDB (a unit for a HiRDB/Parallel Server) during system switchover has exceeded the server failure monitoring time. The server failure monitoring time referred to here is the time specified for the `patrol` operand of the Hitachi HA Toolkit Extension.

For details about the `patrol` operand of the Hitachi HA Toolkit Extension, see the manual *Hitachi HA Toolkit*.

`Y`:

Switches the system without waiting for internal termination processing of HiRDB when internal termination processing of HiRDB during system switchover has exceeded the server failure monitoring time. In this case, system switchover is carried out by assuming that HiRDB has slowed down.

If you are using the standby-less system switchover (1:1) facility or the standby-less system switchover (effects distributed) facility, the specification for this operand is invalid during planned system switchover.

`N`:

Does not switch the system until internal termination processing of HiRDB during system switchover is terminated.

Advantage

If internal termination processing of HiRDB during system switchover takes a long time because, for example, of a disk error, system switchover might be delayed as a result. If you specify `Y` (default value) for this operand, you can switch the system without waiting for internal termination processing of HiRDB, even when it is taking a long time.

Notes

- If you specify `Y` for this operand when the `patrol` operand value is small, planned system switchover might turn into system switchover based on slow-down. This is because internal termination processing of HiRDB during planned system switchover exceeds the time specified by the `patrol` operand.
- You need to be careful if `restart` is specified for the `switchtype` operand of the Hitachi HA Toolkit Extension. If `pd_ha_switch_timeout=Y` (default value) is specified, and if internal termination processing of HiRDB exceeds the server failure monitoring time, HiRDB is not started in the system in which the failure occurred. In this case, the system is immediately switched.

  For details about the `switchtype` operand of the Hitachi HA Toolkit Extension, see the manual *Hitachi HA Toolkit*.

182) pd_ha_mgr_rerun = wait | notwait

This operand is applicable only to HiRDB/Parallel Server.

Specifies whether to wait for the completion of start processing of other units before switching the system for the system manager unit.

`wait`:

Waits for the completion of start processing of other units before switching the system for the system manager unit. In this case, the following checks are performed to start the system manager unit:

- Version check for each unit
- Is the dictionary server running?
- Is at least one front-end server running?
- Is at least one back-end server running?

If another unit has stopped, system switchover for the system manager unit might take a long time or fail. The following table describes what happens when system switchover for the system manager unit occurs when some of the units are stopped.

Reduced activation specification (pd_start_level value)	Specification of a name for the unit that does not start	Action
Not specified (`0`)	--	Cannot start.
Specified (`1`)	Nothing is specified for the `pd_start_skip_unit` operand.	System switchover occurs after a wait time specified by the `pd_reduced_check_time` operand.#
	Stopped unit is specified for the `pd_start_skip_unit` operand.	System switchover is completed immediately.#

Legend:

--: Not applicable

#: System switchover is completed only when all of the following conditions are satisfied:

- The dictionary server is running.
- At least one front-end server is running.
- At least one back-end server is running.

`notwait`:

Does not wait for the completion of start processing of other units before switching the system for the system manager unit. When this operand is specified, system switchover is executed rapidly for the system manager unit without waiting for other units that have stopped to start.

When system switchover is completed for the system manager unit, the message `KFPS05210-I` (system initialization completion message) is issued, even if the HiRDB operating environment is not ready. If a job cannot be continued even after the completion of the system manager switchover because of a UAP error, for example, use the `pdls` command to check the system operating status.

Condition

To specify `notwait` for this operand, the system configuration must satisfy a certain condition. For details about the required system configuration, see the *HiRDB Version 9 System Operation Guide*.

Specification guidelines

To prevent failure of the system switchover for the system manager unit, specify `notwait`. For details about how to handle system manager unit switchover failures, see the *HiRDB Version 9 System Operation Guide*.

183) pd_ha_transaction = <u>error</u> | queuing

Specifies whether to use the transaction queuing facility. Also specifies the processing that takes place when the number of connections to the HiRDB server exceeds the maximum number of concurrent connections (value specified by the `pd_max_users` operand) during system switchover. For details about for the transaction queuing facility, see the *HiRDB Version 9 System Operation Guide*.

`error`:

- Does not use the transaction queuing facility. The transactions being processed by the back-end server or dictionary server being switched end in errors.
- If the number of connections to the HiRDB server exceeds the maximum number of concurrent connections during system switchover, the connections to the HiRDB server end in errors.

`queuing`:

- Uses the transaction queuing facility. Instead of ending in errors, the transactions being processed by the back-end server or dictionary server being switched are queued by the front-end server until system switchover is completed. SQL response time will be longer than normal in this case because processing will wait until the back-end server and dictionary server units have started.
- If the number of connections to the HiRDB server exceeds the maximum number of concurrent connections during system switchover, connection processing to the HiRDB server is retried for the duration of `pd_ha_trn_queuing_wait_time` + `pd_ha_trn_restart_retry_time`. However, the HiRDB client version must be 07-00 or later.

Conditions

If you use the transaction queuing facility, all of the following conditions must be satisfied:

- HiRDB/Parallel Server
- The rapid system switchover facility, standby-less system switchover (1:1) facility, or standby-less system switchover (effects distributed) facility is used.

Note that these conditions need not be satisfied if connection processing to the HiRDB server is to be retried by a HiRDB client.

Relationship to client environment definition

Even when `queuing` is specified for this operand, you can specify that the transaction queuing facility not be used for each client. To cancel the transaction queuing facility for each client, specify the `PDHATRNQUEUING` operand in the client environment definition. For details about the `PDHATRNQUEUING` operand, see the *HiRDB Version 9 UAP Development Guide*.

Relationship to other operands

This operand is related to the following operands:

- `pd_ha_trn_queuing_wait_time`
- `pd_ha_trn_restart_retry_time`

184) pd_ha_trn_queuing_wait_time = *transaction-queuing-wait-time*

~<unsigned integer>((1-3600))<<180>>(seconds)

Specifies the transaction queuing wait time when the transaction queuing facility is used. If the standby unit or server does not start within the wait time specified by this operand, the transactions being queued end in errors. Transactions that occur subsequently end in errors without being queued. If a unit or server starts before this wait time, transactions are resumed without waiting for the time specified by this operand.

If the standby-less system switchover (1:1) facility is used, the alternate portion becomes the standby system during normal operation, and the normal BES unit becomes a standby system during alternation.

Condition

The `pd_ha_transaction` operand must be set to `queuing`.

Specification guidelines

- Normally, you need not specify this operand. However, if the rollforward during system switchover takes 180 seconds or longer, increase the value of this operand.
- Specify this value taking into consideration the time required for switchover. When executing a UAP that has a long transaction processing time, the time required for rollforward processing at switchover must also be considered.
- Choose a value for this operand that reflects the time required for disk switching during switchover. If the specified value is large, it will take longer for the transaction to detect the error if switchover fails. If the value is small, a transaction error might result even when switchover is successful.

185) pd_ha_trn_restart_retry_time = *retry-time-upper-limit-after-transaction-start-request-errors*

~<unsigned integer>((1-3600))<<60>>(seconds)

If system switchover occurs while the transaction queuing facility is being used, transactions are queued by the front-end server. However, during the period between the system switchover and the restart of the standby unit or server, the front-end server cannot detect the system switchover. During this period (the period between the system switchover and the restart of the standby unit or server), the front-end server issues a transaction start request to the running unit or server. However, because the running unit has already been abnormally terminated, this transaction start request ends in an error. For the transaction that ends in an error, the front-end server re-issues a transaction start request (retries the transaction start request).

This operand specifies the upper limit for this retry time. If the standby unit or server is not restarted within the value specified by this operand, the transactions being retried end in errors. Furthermore, transactions that occur subsequently end in errors without being retried. Note that if the standby unit or server begins to restart before this retry time, no retries occur and transactions are queued.

Condition

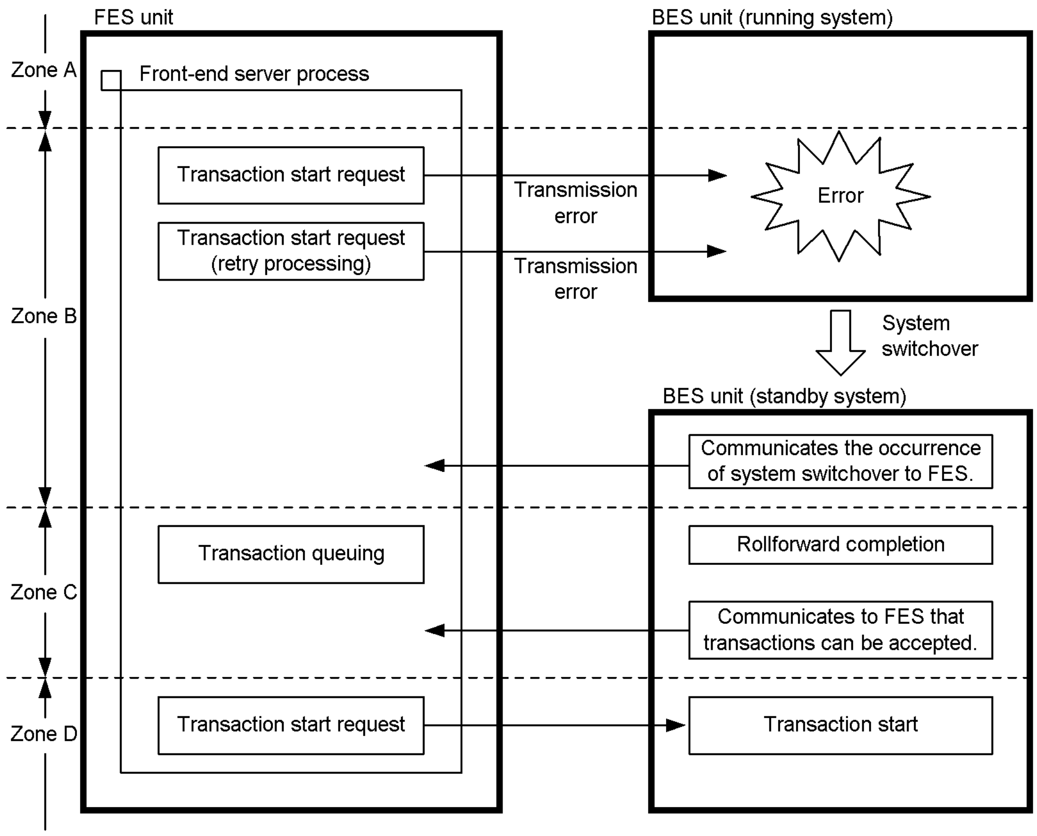
The `pd_ha_transaction` operand must be set to `queuing`.

Specification guidelines

- Normally, you need not specify this operand. However, if disk switching by cluster software takes 60 seconds or longer, increase the value of this operand.
- Choose a value for this operand that reflects the time required for disk switching during switchover. If the specified value is large, it will take longer for the transaction to detect the error if switchover fails. If the value is small, a transaction error might result even when switchover is successful.

Remarks

The relationship between the `pd_ha_trn_queuing_wait_time` and `pd_ha_trn_restart_retry_time` operands is explained as follows.


FES unit
BES unit (running system)
Zone A
Front-end server process
Transaction start request
Transaction start request (retry processing)
Transmission error
Transmission error
Error
Zone B
System switchover
BES unit (standby system)
Communicates the occurrence of system switchover to FES.
Transaction queuing
Rollforward completion
Zone C
Communicates to FES that transactions can be accepted.
Zone D
Transaction start request
Transaction start

Explanation:

Zones A and D:

Transactions can be started (normal state).

Zone B:

The unit containing the back-end server is being switched, and the front-end server cannot detect this condition. Transaction start request is retried for the duration of time specified by the `pd_ha_trn_restart_retry_time` operand. When the front-end server detects the system switchover, transactions are queued. If the front-end server cannot detect it within the specified time, the transactions end in errors.

Zone C:

The unit containing the back-end server is being switched, and the front-end server has detected this condition. Transactions are queued for the duration of time specified by the `pd_ha_trn_queuing_wait_time` operand. If the transactions cannot be started within the specified time, they end in errors.

186) pd_ha_resource_act_wait_time = *maximum-wait-time-for-resource-activation*

~<unsigned integer>((2-3600))<<10>> (seconds)

This operand specifies the maximum wait time for the running server's resources to be activated at the time of unit startup when you use the standby-less system switchover (effects distributed) facility. Unit startup processing is placed on hold up to the specified amount of wait time. If the resources are activated within the specified amount of time, unit startup processing resumes.

Advantages

When unit startup processing is completed, jobs can be started only if the running server's startup processing is completed in the unit. By specifying an appropriate value in this operand, you can start your jobs immediately after the unit's startup processing is completed because the unit startup processing will have waited on wait status for resource activation.

Specification guidelines

Normally, there is no need to specify this operand. Specify this operand when all the following conditions are applicable:

- The standby-less system switchover (effects distributed) facility is being used
- The `KFPS05623-I` message was displayed
- The target unit for the message contains the running server

Use the following guideline to determine the value to be specified in this operand:

10 + *time required for resource activation processing* (seconds)

Remarks

If the unit does not contain the running server, the running server's startup is placed on hold for the amount of time specified in this operand. However, if all the servers in the unit start as standby servers, unit startup processing is restarted without waiting for the amount of time specified in this operand.

187) pd_deter_restart_on_stop_fail = Y | N

When switchover is being run in the server mode, specifies whether switchover is to be suppressed when a unit terminates abnormally during forced stop processing by the `pdstop -f` or `pdstop -z` command in the primary system.

`Y`:

Suppresses switchover when a unit terminates abnormally during forced stop processing.

`N`:

Does not suppress switchover when a unit terminates abnormally during forced stop processing.

Note

To use this facility, the version of the Hitachi HA Toolkit Extension combined with HiRDB must be 01-20 or later. When combined with a product whose version is earlier than 01-20, specifying `Y` in this operand is ignored and `N` is assumed to have been specified.

2.2.35 Operands related to HiRDB Datareplicator

188) pd_rpl_init_start = Y | N

Specifies whether the HiRDB Datareplicator linkage facility is to be used from the time of HiRDB startup.

`Y`: Use the HiRDB Datareplicator linkage facility from the time of HiRDB startup.

`N`: Do not use the HiRDB Datareplicator linkage facility from the time of HiRDB startup.

Operation method

- When `N` is specified for this operand, the HiRDB Datareplicator linkage facility can be used by entering the `pdrplstart` command.
- The `pdrplstop` command is entered to stop the HiRDB Datareplicator linkage facility.
- When data linkage has been stopped by entering the `pdrplstop` command, HiRDB Datareplicator linkage can be restarted by entering the `pdrplstart` command. However, before the `pdrplstart` command is entered, the target database must be re-created based on the extracted database.

189) pd_rpl_reflect_mode = server | uap

This operand is applicable only to HiRDB/Parallel Server.

Specifies `uap` in order to use HiRDB Datareplicator's target transaction synchronization facility. If you do not use the target transaction synchronization facility, there is no need to specify this operand. For details about the target transaction synchronization facility, see the *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide*.

`server`: Incorporates transactions for each HiRDB server.

`uap`: Incorporates transactions for each UAP.

Condition

When you specify this operand, you must also specify the `pd_rpl_hdepath` operand. If the `pd_rpl_hdepath` operand is omitted, this operand is ignored.

Notes

When `uap` is specified, the amount of update log information increases. In such a case, refer to the *HiRDB Version 9 Installation and Design Guide* and re-estimate the total size of the system log files.

Relationship to other operands

If you specify `uap` in this operand, re-estimate the value of the `pd_log_max_data_size` operand.

Effects on individual estimation formulas

If the value of the `pd_rpl_reflect_mode` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the size of system log files*

190) pd_log_rpl_no_standby_file_opr = <u>stop</u> | continue

Specifies the action to be taken when a swap request is received when not all of the system log files can be created at the swapping destination because system log extraction by HiRDB Datareplicator using the HiRDB Datareplicator linkage facility has not been completed.

`stop`: Stop the HiRDB unit forcibly.

`continue`: Stop HiRDB Datareplicator linkage and continue operation with HiRDB only.

Condition

Specification of this operand is valid only if `Y` is specified for the `pd_rpl_init_start` operand or if the `pdrplstart` command has been entered and the HiRDB Datareplicator linkage facility is being used. In the case of a HiRDB/Parallel Server, this operand is not valid for front-end servers and dictionary servers.

Notes

- When `stop` (the default value) is specified for this operand and the HiRDB unit is terminated forcibly, the HiRDB unit must be restarted after it has been confirmed that the log file being extracted on the HiRDB Datareplicator side has moved to another file (meaning that at least one file has been extracted completely). Even if the HiRDB unit is restarted immediately after the forced termination, HiRDB will recognize the abnormal status and forcibly terminate the HiRDB unit again if there is no swap target file.
- When `continue` is specified for this operand and HiRDB Datareplicator linkage is stopped, a mismatch will occur between the extracted database and the target database on the HiRDB Datareplicator side. Therefore, it is necessary to re-create the target database.

191) pd_rpl_func_control = BACKWARD_CUTOFF_UPDATE | <u>NONE</u>

~<identifier>

Specifies the facility to be run by the source HiRDB when the HiRDB Datareplicator linkage facility is used.

`BACKWARD_CUTOFF_UPDATE`:

Performs backward cutoff/update of `BLOB` data and `BINARY` data. For details about backward cutoff/update of `BLOB` and `BINARY` data, see the *HiRDB Version 9 UAP Development Guide*.

`NONE`:

Does not perform backward cutoff/update of `BLOB` data and `BINARY` data.

Specification guidelines

- `BACKWARD_CUTOFF_UPDATE` can be specified as follows.

pd_rpl_hdepath operand specified in unit control information definition?	Source HiRDB Datareplicator version	BACKWARD_CUTOFF_UPDATE specification
Yes	08-01 or later	#1
	08-00 or earlier	Not permitted#2
No	--	Permitted

Legend:

--: Not applicable

#1: See *Data linkage in tables that use BLOB and BINARY types* in the manual *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide*.

#2: Specify NONE for this operand.

If the conditions above are not satisfied, do not specify BACKWARD_CUTOFF_UPDATE. If it is specified anyway, the results of backward cutoff/update of BLOB and BINARY data in an extracted table in the source database will not be imported into the target database.

- If the pd_rpl_hdepath operand is specified and the version of the source Datareplicator is 08-00 or earlier, specify NONE for this operand.

Notes

- Identifiers might be specified using either upper-case or lower-case letters.
- If an identifier is specified more than once, the result is the same as if it had been specified only once.
- If NONE is specified after BACKWARD_CUTOFF_UPDATE had been specified, check whether all the update logs were completely imported prior to the change in the specification. If the operand specification is modified while there remain update logs that have not been imported, problems might arise, such as SQL execution counts that do not match between the source and target databases.
- If NONE is specified after a stored procedure, stored function, or trigger was defined and BACKWARD_CUTOFF_UPDATE had been specified, re-create ALTER PROCEDURE, ALTER ROUTINE, or ALTER TRIGGER, respectively. If you do not, an error will result when the stored procedure, stored function, or trigger is executed. However, there is no need to re-create if no backward cutoff/update of BLOB or BINARY data had yet been performed. For details about ALTER PROCEDURE, ALTER ROUTINE, and ALTER TRIGGER, see the manual *HiRDB Version 9 SQL Reference*.
- If BACKWARD_CUTOFF_UPDATE is specified after a stored procedure, stored function, or trigger has been defined and NONE had been specified, re-create ALTER PROCEDURE, ALTER ROUTINE, or ALTER TRIGGER, respectively. Not re-creating will not result in an error, but backward cutoff/update of BLOB and BINARY data will not be performed.

2.2.36 Operands related to linkage to JP1

192) pd_jp1_use = Y | N

Specifies whether events such as HiRDB startup and termination are to be registered into JP1. For details about HiRDB events that can be registered, see the *HiRDB Version 9 Installation and Design Guide*.

Y: Registers HiRDB events in JP1.

N: Does not register HiRDB events in JP1.

Note

Windows (x64) does not support event notification to JP1. If Y is specified in this operand for Windows (x64), the KFPS04605-W message is displayed during HiRDB startup.

193) pd_jp1_event_level = 1 | 2

Specifies the HiRDB event types (basic or extended attribute) that are registered in JP1. For details about basic and extended attributes, see the *HiRDB Version 9 Installation and Design Guide*.

1: Registers only the basic attributes.

2: Registers both the basic and extended attributes.

Condition

This operand is valid when `Y` is specified for the `pd_jp1_use` operand.

Specification guidelines

- Extended attributes can be registered only when the HiRDB version is 07-02 or later.
- In the case of Windows (x64), this operand cannot be specified.

194) pd_jp1_event_msg_out = Y | N

Specifies whether to register message-outputting events in JP1.

`Y`: Registers message-outputting events.

`N`: Does not register message-outputting events.

Condition

This operand is valid when `Y` is specified for the `pd_jp1_use` operand.

Specification guidelines

Specify this operand if you want to reduce the number of registrations in JP1 when the volume of messages output by HiRDB is large.

2.2.37 Operands related to OLTP

195) pd_oltp_holdcr = use | nouse

Specifies whether to use the holdable cursor facility in a UAP under the OLTP environment.

`use`: Uses the holdable cursor facility in a UAP under the OLTP environment.

`nouse`: Does not use the holdable cursor facility in a UAP under the OLTP environment.

Specification guidelines

Specify `use` for this operand only when you plan to use the holdable cursor facility in a UAP under the OLTP environment.

Note

To use the holdable cursor facility in a UAP under the OLTP environment, you need to use a UAP that satisfies a certain condition. For details about the holdable cursor, see the *HiRDB Version 9 UAP Development Guide*.

2.2.38 Operands related to version upgrade

196) pd_auto_vrup = Y | N

Specifies whether the `pdvrup` command for executing HiRDB version upgrading is to be started automatically. For details about how to upgrade the HiRDB version, see the *HiRDB Version 9 Installation and Design Guide.*

`Y`: Start the command automatically.

`N`: Do not start the command automatically (the HiRDB administrator must enter the `pdvrup` command).

197) pd_sysdef_default_option = recommendable | v6compatible | v7compatible

Specifies use of a previous version's default values. This operand's default value depends on the version.

`recommendable`: Applies the default values of the most recent version.

`v6compatible`: Applies the default values of HiRDB Version 6 or earlier.

`v7compatible`: Applies the default values of HiRDB Version 7.

Specification guidelines

Normally, omit this operand. This operand is used when problems arise from changing to the default values set by upgrading. For details about the applicable operands, see *1.5(1) Operands whose default value depends on the version*.

2.2.39 Operands related to communication processing

198) pd_service_port = ***scheduler-process-port-number***

~<unsigned integer>((5001-65535))

Specifies the port number for the scheduler process under the following circumstances:

- A high-speed connection facility is used

  For details about high-speed connection facilities, see the *HiRDB Version 9 Installation and Design Guide.*
- A firewall or NAT is installed on the HiRDB server

  For details about setting the HiRDB environment when a firewall or NAT is installed on the HiRDB server, see the *HiRDB Version 9 Installation and Design Guide*.

Specification guidelines

- Use this operand for a HiRDB/Single Server or when there is a firewall between the HiRDB client and the HiRDB server.
- Use this operand in systems connected via a broadband LAN to reduce the number of communications when connecting to a server.

Notes

- If there are multiple units in a single server machine, such as in a mutual system switchover configuration, always specify this operand in the unit control information definition. Specify a different port number for each unit. If a port number is duplicated, system switchover will fail in one of the units during a switching attempt.
- When this operand is specified in both the unit control information definition and the system common definition, the specification in the unit control information definition is effective.
- If the HiRDB port number (the port number specified with the `pd_name_port` operand or the `-p` option of the `pdunit` operand) is specified, the specification of this operand will be ignored, which will prevent operation using the high-speed connection facility and when a firewall or NAT is set up at the HiRDB server. The `KFPS00860-W` message will be issued when this happens.
- If the high-speed connection facility is used to issue concurrently more connection requests than the value set in the `pd_max_users` operand, a shortage will occur in the number of active front-end servers and single servers for extracting connection requests from the message queue. Consequently, some connection requests will not be extracted from the message queue, and the message queue monitoring facility could stop the unit. You must ensure that the number of concurrent connection requests generated does not exceed the value specified in the `pd_max_users` operand. For details about the message queue monitoring facility, see the *HiRDB Version 9 System Operation Guide*.
- For details about how to specify the scheduler process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.
- In the case of a parallel server that has multiple front-end servers, the front-end servers to be connected are fixed and load sharing will no longer be possible. For this reason, consider carefully the balance among the front-end servers to be connected.

Relationship to other operands

This operand is related to the following operands:

- `pd_scd_port`
- `pdunit -s`

199) pd_name_fixed_port_lookup = Y | <u>N</u>

This operand is applicable only to HiRDB/Parallel Server.

Specifies how addresses are to be resolved for communication between units that is handled by the system server. Address resolution refers to the processing that acquires the IP address and port number of a system server.

`Y`: Resolve addresses in the local units' shared memory.

`N`: Resolve addresses by communicating with all units (broadcast).

When `Y` is specified, the operands shown below must be specified and the port number of the system server that communicates with the units must be specified. If the port number is not specified, HiRDB cannot start. For details about how to specify port numbers and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Port type	Operand
HiRDB port	`pd_name_port` operand or `-p` option of `pdunit` operand
Transaction server process port	`pd_trn_port` operand or `-t` option of `pdunit` operand
Scheduler process port	`pd_scd_port` operand or `-s` option of `pdunit` operand
Unit monitoring process port#1	`pd_alv_port` operand or `-a` option of `pdunit` operand
Message log server process port#2	`pd_mlg_port` operand or `-m` option of `pdunit` operand

#1: In single servers and in units that do not have a system manager, the unit monitoring process does not start, so it is not necessary to specify a port number.

#2: When `local` is specified in the `pd_mlg_msg_log_unit` operand, broadcast does not occur, so it is not necessary to specify a port number. Also, when `manager` is specified in the `pd_mlg_msg_log_unit` operand, the message log server process does not start in units that do not have a system manager, so it is not necessary to specify a port number.

Advantage

When `Y` is specified in this operand and the port number of the system manager that communicates with units is specified, broadcast does not occur when an address is resolved. This avoids the following problems when there is a large number of units:

- Pressure on communication resources (high network workload and use of all available ports)
- Decline in performance

Notes

- When `N` is specified in this operand, broadcasts to all units occur as a means of resolving addresses. Broadcasting occurs in the following cases:
  ● When an address is resolved initially.
  ● When an address is resolved after the system server that previously resolved the address has been taken out of service.
  ● When an address is resolved after a unit of the system server that previously resolved the address has been taken out of service.
- There is no communication between units in the case of a HiRDB/Single Server or in the case of a HiRDB/Parallel Server consisting of only one unit, so such a server functions as though `N` were specified, regardless of the actual specification of this operand.
- It is advisable to use a high-speed connection for the client connection format. If you use a normal connection, a broadcast might occur briefly when the client is connected. For details about client connection formats, see the *HiRDB Version 9 UAP Development Guide*.

200) pd_scd_port = *scheduler-process-port-number*

~<unsigned integer>((5001-65535))

Specifies the port number of the scheduler process under the following circumstances:

- `Y` is specified in the `pd_name_fixed_port_lookup` operand
- A high-speed connection facility is used
  For details about high-speed connection facilities, see the *HiRDB Version 9 UAP Development Guide*.
- A firewall or NAT is set up on the HiRDB server
  For details about the HiRDB environment settings in such a case, see the *HiRDB Version 9 Installation and Design Guide*.

Notes

- To specify a different port number for each unit, use the `-s` option of the `pdunit` operand.
- For details about how to specify a scheduler process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.

- This operand is related to the following operands:
  - `pd_service_port`
  - `pdunit -s`

201) pd_trn_port = *transaction-server-process-port-number*

~<unsigned integer>((5001-65535))

When `Y` is specified in the `pd_name_fixed_port_lookup` operand, specifies the port number of the transaction server process.

Notes

- For details about how to specify a transaction server process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.
- This operand is related to the `-t` option of the `pdunit` operand.

202) pd_mlg_port = *message-log-server-process-port-number*

~<unsigned integer>((5001-65535))

When `Y` is specified in the `pd_name_fixed_port_lookup` operand, specifies the port number of the message log server process

Notes

- Port numbers are not used for units on which the message log server process does not start, but a check is performed that the defined port number does not duplicate any other port number. In a HiRDB/Parallel Server consisting of at least two units, the message log server process does not start on units that do not have a system manager when `manager` is specified for `pd_mlg_msg_log_unit`.
- For details about how to specify a message log server process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.
- This operand is related to the `-m` option of the `pdunit` operand.

203) pd_alv_port = *unit-monitoring-process-port-number*

~<unsigned integer>((5001-65535))

This operand is applicable only to HiRDB/Parallel Server.

When `Y` is specified in the `pd_name_fixed_port_lookup` operand, specifies the port number of the unit monitoring process.

Notes

- Port numbers are not used for units on which the unit monitoring process does not start, but a check is performed that the defined port number does not duplicate any other port number. The unit monitoring process does not start on any unit in a HiRDB/Parallel Server consisting of at least two units, except on units that have a system manager.
- For details about how to specify a unit monitoring process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.
- This operand is related to the `-a` option of the `pdunit` operand.

204) pd_change_clt_ipaddr = 0 | <u>1</u>

Specifies the network to be used by the HiRDB server to communicate with HiRDB clients. Normally, this operand need not be specified.

`0`:

Communication from the HiRDB server to HiRDB clients uses the network in which the IP address specified in the `PDCLTRCVADDR` operand of the client environment definition is located. If the `PDCLTRCVADDR` operand is omitted, the IP address of the standard host is assumed.

`1`:

For communication from the HiRDB server to HiRDB clients, the network used for communication from HiRDB clients to the HiRDB server is used.

Relationship to other operands

If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `0`.

205) pd_registered_port =*"port-number-reservation-range"* [,*"port-number-reservation-range"*]...

~<character string>

Specifies ranges of port numbers to be used for communication by the HiRDB server when the HiRDB reserved port facility is used.

The HiRDB reserved port facility is enabled for server-to-server and server-to-client communications. For details, see the description of the `pd_registered_port_level` operand. This facility does not need to be used if the number of ports to be used is small.

Operand specification example

This example provides the following 35,000 port numbers: 6000 to 8999, 12500 to 29999, and 30500 to 44999:

```
set pd_registered_port = "6000:8999","12500:29999","30500:44999"
```

Advantages

The port numbers to be used for communication between HiRDB servers are assigned automatically by the OS. If the communication volume increases significantly, a shortage of port numbers might cause an interruption of processing or might adversely affect the communication processing by other programs. Use this operand to specify a range of port numbers to be used exclusively by HiRDB to prevent such problems.

Specification guidelines

- The range of port numbers that can be specified is from 5001 through 49151.
- For the number of ports used by HiRDB, see the *HiRDB Version 9 Installation and Design Guide*.
- Do not register the port number specified in this operand in `%windir%\system32\drivers\etc\services` (definition location in a DNS environment). A duplication check is performed for `%windir%\system32\drivers\etc\services` when `Y`, `C`, or `W` is specified in the `pd_registered_port_check` operand.

Specification value tuning method

As a guideline, the value obtained from the following formula is the total number of ports that will be needed:

a + b + 100

a: Number of HiRDB reserved ports that are being used (`# OF REGISTERED PORTS`)

b: Number of ports being used that were assigned automatically by the OS because all HiRDB reserved ports were in use (`# OF ASSIGNED PORTS`)

These values can be determined from the statistical information related to system operation obtained by executing the statistics analysis utility.

Operand rules

- Up to 10 ranges of port numbers can be specified.
- If more than one range is specified, ensure that there is no overlap in the port numbers included in the various ranges.
- An ending port number must be greater than the paired starting port number.

Notes

- If there are multiple units on a single server machine, such as in a mutual system switchover configuration, this operand must be specified in the unit control information definition. Be sure to specify a different port number for each unit. If a port number is duplicated, system switchover will fail in one of the units during a switching attempt.
- Specification of a port number that is not supported by the HiRDB reserved port facility is ignored. For the scope of the HiRDB reserved port facility, see the `pd_registered_port_level` operand.
- For details about how to specify port numbers when the HiRDB reserved port facility is used and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

- When this operand is specified in both the unit control information definition and the system common definition, the specification in the unit control information definition is effective.

Effects on individual estimation formulas

If the value of the `pd_registered_port` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

206) pd_registered_port_check = Y | N | C | W

Specifies whether checking is to be performed for overlapping port numbers in the ranges of port numbers specified in the `pd_registered_port` operand and in the port numbers registered in `%windir%\system32\drivers\etc\services` (definition location in a DNS environment).

`Y`:

Check for overlap. If an overlap is found, the `KFPS00348-E` message is output and HiRDB activation is canceled.

`N`:

Do not check for overlap.

`C`:

Check for overlap. The HiRDB reserved port facility is not applied to any overlapping port number.

`W`:

Check for overlap. If an overlap is found, the `KFPS00354-W` message is output. The HiRDB reserved port facility is not applied to any overlapping port number.

Condition

The `pd_registered_port` operand must be specified.

Specification guidelines

- If an overlap in port numbers is detected, HiRDB communication might be adversely affected, resulting, for example, in the receipt of wrong messages or message send failures.
- If `Y`, `C`, or `W` is specified, process server process activation in a DNS environment might slow down.

207) pd_registered_port_level = 0 | 1

Specifies the target range for the HiRDB reserved port facility.

Condition

The `pd_registered_port` operand must be specified.

Specification guidelines

The following shows the target range depending on the value of this operand (0 or 1):

Port numbers used by HiRDB		Value of pd_registered_port_level	
		0	1
Ports used by HiRDB server	Send port number for server-to-server communication	Y	Y
	Receive port number for server-to-server communication	Y	Y
	Port number specified in `pd_name_port` operand	N	N
	Port number specified in `-p` option of `pdunit` operand	N	N
	Port number specified in `pd_service_port` operand	N	N
	Port number specified in `pd_scd_port` operand	N	N
	Port number specified in `-s` option of `pdunit` operand	N	N

Port numbers used by HiRDB		Value of pd_registered_port_level	
		0	1
	Port number specified in `pd_trn_port` operand	N	N
	Port number specified in `-t` option of `pdunit` operand	N	N
	Port number specified in `pd_mlg_port` operand	N	N
	Port number specified in `-m` option of `pdunit` operand	N	N
	Port number specified in `pd_alv_port` operand	N	N
	Port number specified in `-a` option of `pdunit` operand	N	N
	Send port number for command	N	N
	Receive port number for command	N	N
	Send port number for client	N	Y
	Receive port number for client	Y	Y
	Service port number for communication port	N	N
	Port number used for lock control	N	N
Ports used by HiRDB client	Send port number	N	N
	Receive port number	N	N

Legend:

Y: Subject to the HiRDB reserved port facility. A port number specified in the `pd_registered_port` operand is used.

N: Not subject to the HiRDB reserved port facility.

208) pd_ipc_send_retrycount = ***process-to-process-send-retries-count***

~<unsigned integer>((1-32767))<<200>>

Specifies the number of times process-to-process communication can be attempted. This operand is related to the `pd_ipc_send_retrysleeptime` operand.

Examples

- `pd_ipc_send_retrycount = 500`
- `pd_ipc_send_retrysleeptime = 2`

When the operands are specified in this way, send is attempted up to 500 times and a 2-second sleep occurs between attempts.

Specification guidelines

- Normally, this operand need not be specified.
- Specifying too large a value for this operand increases the CPU lock rate.

209) pd_ipc_send_retrysleeptime = ***process-to-process-send-retry-sleep-time***

~<unsigned integer>((0-60)) <<0>> (seconds)

Specifies the sleep time between process-to-process send retries.

This operand is related to the `pd_ipc_send_retrycount` operand.

Examples

- `pd_ipc_send_retrycount = 500`
- `pd_ipc_send_retrysleeptime = 2`

When the operands are specified in this way, send is attempted up to 500 times and a 2-second sleep occurs between attempts.

Specification guidelines

- Normally, this operand need not be specified.
- Specifying too large a value for this operand increases communication completion time.

210) pd_ipc_send_count = ***server-to-server-send-retries-count***

~<unsigned integer>((1-32767)) <<60>>

Specifies the number of times a server-to-server send can be performed before the send is completed. A send occurs for up to 5 seconds at each retry. With the default value, send will be performed for up to 60 × 5 seconds = 300 seconds.

Specification guideline

Normally, this operand need not be specified. If transmission timeouts occur frequently, increase this operand's value.

211) pd_ipc_recv_count = ***server-to-server-receive-retries-count***

~<unsigned integer>((1-32767)) <<120>>

Specifies the number of times a server-to-server receive can be performed before receive is completed. Receive occurs for up to 5 seconds at each retry. With the default value, receive is performed for up to 120 × 5 seconds = 10 minutes.

Specification guideline

Normally, this operand need not be specified.

212) pd_ipc_conn_nblock = Y | N

Specifies whether to establish connections in the non-block mode for HiRDB server-to-server communication.

`Y`: Establishes the connection in the non-block mode.

`N`: Establishes the connection in the block mode.

Advantage

When you specify `Y` for this operand, you can issue `connect()` system calls in the non-block mode for HiRDB server-to-server communication . As a result, the duration for which `connect()` system calls block processes during a LAN error, which used to last dozens of seconds (depending on the OS), can now be shortened to the time specified in the `pd_ipc_conn_nblock_time` operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_ipc_conn_nblock_time`
- `pd_ipc_conn_interval`
- `pd_ipc_conn_count`
- `pd_host_watch_interval`

Specification guidelines

Normally, there is no need to specify this operand.

Remarks

To establish a communications connection between a client and a server, specify the necessary parameter in the `PDNBLOCKWAITTIME` or `pd_ipc_clt_conn_nblock` operand in the client environment definition. (`PDNBLOCKWAITTIME` operand).

213) pd_ipc_conn_nblock_time = ***connection-establishment-monitoring-time-in-non-block-mode***

~<unsigned integer>((1-120))<<8>>(seconds)

Specifies the retry interval and monitoring time for establishing connections in the non-block mode for HiRDB server-to-server communication.

Condition

This operand is valid when `Y` is specified for the `pd_ipc_conn_nblock` operand.

Advantage

If unreasonable communication errors occur frequently, increasing the value specified for this operand might reduce the number of communication errors.

Specification guidelines

Normally, there is no need to specify this operand.

If unreasonable communication errors occur frequently, use the `ping` command of the OS, for example, to measure the communication time between HiRDB servers. If that time is greater than the value specified for this operand, specify a value greater than that communication time for this operand.

Note

- Increasing the specification value for this operand increases the timeout duration when communication is disabled. Consequently, other processes might be adversely affected.
- For this operand, do not specify a value that is greater than the OS's wait time for establishing a connection in the block mode. If a larger value is specified, connection times out when the OS's wait time is reached.

Relationship to other operands

This operand is related to the following operands:

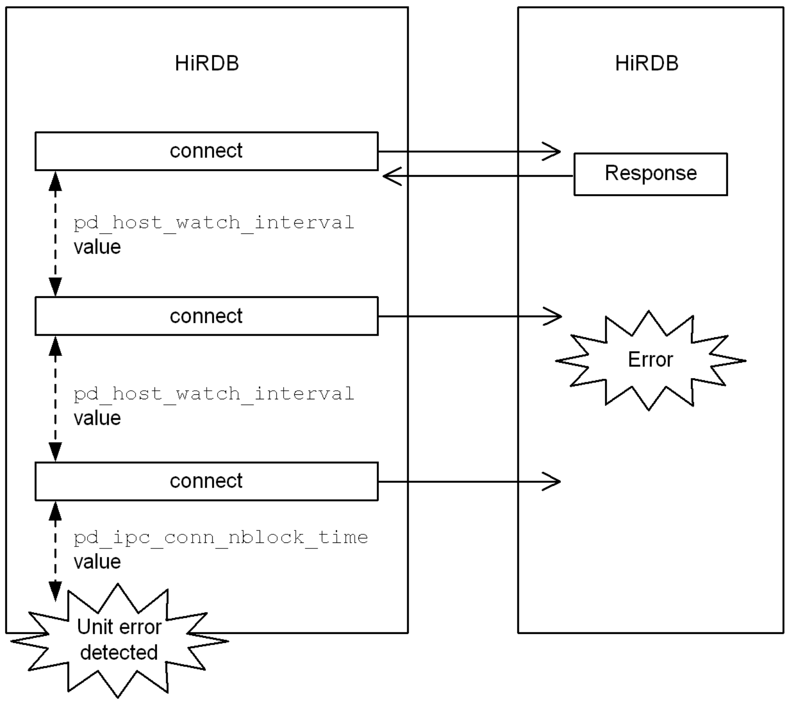
- `pd_ipc_conn_nblock`
- `pd_ipc_conn_interval`
- `pd_ipc_conn_count`

Remarks

HiRDB communicates with other units at a given interval (the value of the `pd_host_watch_interval` operand; the default value is 10 seconds) to monitor the operating status of other units.

If a LAN error occurs immediately after a monitoring-target unit has responded, HiRDB tries to communicate with that unit after a certain wait time (the value of the `pd_host_watch_interval` operand). Thereafter, communication retry is carried out for the duration specified by the `pd_ipc_conn_nblock_time` operand (the default value is 8 seconds). Therefore, the time until the abnormal termination of the monitoring-target unit is detected is the time combining the values of the two operands.

The following figure shows the time line until HiRDB detects the abnormal termination of the monitoring-target unit.


HiRDB
HiRDB
connect
Response
pd_host_watch_interval value
connect
Error
pd_host_watch_interval value
connect
pd_ipc_conn_nblock_time value
Unit error detected

214) pd_ipc_clt_conn_nblock = Y | N

Specifies whether to establish connections in the non-block mode for communication with HiRDB clients.

`Y`: Establishes the connection in non-block mode. Connection establishment processing is monitored at the interval specified in for `pd_ipc_clt_conn_nblock_time` operand.

`N`: Establishes the connection in block mode. This is the same operation supported in versions up to 09-02.

Advantages

When you specify `Y` for this operand, you can issue `connect()` system calls in non-block mode for communication with the HiRDB client. As a result, the duration for which `connect()` system calls block processes during a LAN error, which used to last dozens of seconds (depending on the OS), can now be shortened to the time specified in the `pd_ipc_clt_conn_nblock_time` operand.

Specification guidelines

Normally, this operand does not need to be specified.

Relationship to the client environment definition

The value of this operand might be ignored depending on the value of `PDNBLOCKWAITTIME` specified in the client environment definition. The following table describes which mode is used to establish a connection depending on the combination of this operand and `PDNBLOCKWAITTIME` in the client environment definition.

Value of PDNBLOCKWAITTIME in the client environment definition	Value of the pd_ipc_clt_conn_nblock operand	
	Y	N
`0`	• When the server establishes a connection with the client: Non-block mode Connection establishment processing is monitored at the interval specified in the `pd_ipc_clt_conn_nblock_time` operand. • When the client establishes a connection with the server: Block mode	Block mode
`1` or greater	Non-block mode Connection establishment processing is monitored at the interval specified for `PDNBLOCKWAITTIME` in the client environment definition.	

215) pd_ipc_clt_conn_nblock_time = ***connection-establishment-monitoring-interval-in-non-block-mode-(communication-with-HiRDB-clients)***

~<unsigned integer> ((1-120))<<8>>(seconds)

Specifies the monitoring interval when establishing a connection with a HiRDB client in the non-block mode.

Condition

This operand only takes effect if `Y` is specified for the `pd_ipc_clt_conn_nblock` operand.

Specification guidelines

Normally, this operand does not need to be specified.

If too many communication errors occur frequently, use a command such as the OS's `ping` command to measure the communication time. If that time is greater than the value specified for this operand, specify a value greater than the observed communication time for this operand.

Notes

For this operand, do not specify a value that is greater than the OS's wait time for establishing a connection in the block mode. If a larger value is specified, connection times out when the OS's wait time is reached.

216) pd_ipc_conn_interval = ***connection-establishment-retry-interval***

~<unsigned integer>((0-50))<<1>>(seconds)

Specifies the retry interval when establishing a connection for sending data between HiRDB servers. Connection establishment retries are made in the following cases:

- A connection timeout has occurred.
- A shortage has occurred in the `Listen` queue.

Specification guidelines

- Normally, there is no need to specify this operand.
- When specifying this operand, make sure that the following condition is satisfied:

This operand value < value of pd_ipc_conn_nblock_time

Relationship to other operands

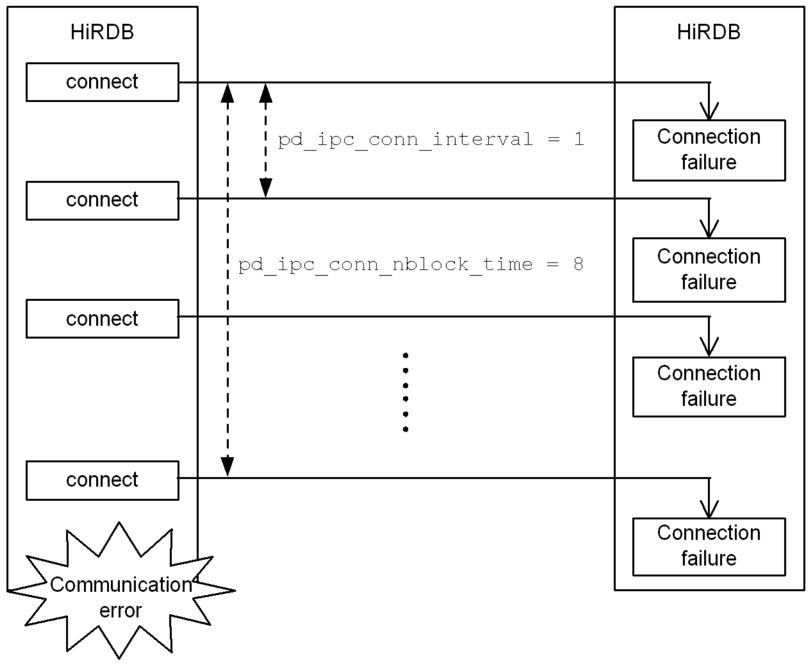
This operand is related to the following operands:

- `pd_ipc_conn_count`
- `pd_ipc_conn_nblock`
- `pd_ipc_conn_nblock_time`

Remarks

The following figure shows examples of retries in non-block mode and block mode communication.

- Retry processing during non-block mode communication (when `pd_ipc_conn_nblock = Y`)

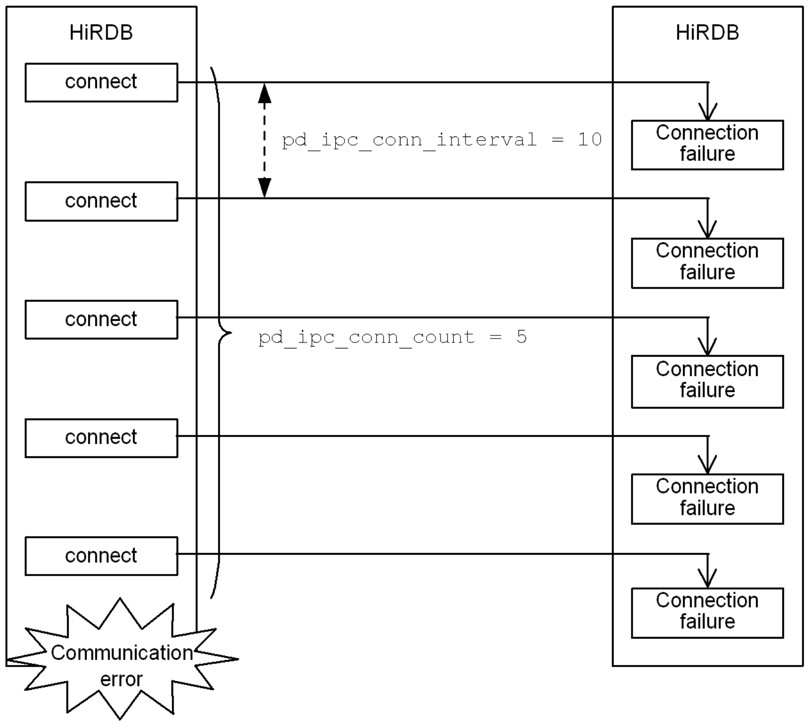


Explanation

With the following specifications, non-block mode connection establishment is retried for 8 seconds at 1-second intervals.

- `pd_ipc_conn_nblock_time` = 8
- `pd_ipc_conn_interval` = 1

- Retry processing during block mode communication (when `pd_ipc_conn_nblock = N`)



Explanation

With the following specifications, non-block mode connection establishment is retried 5 times at 10-second intervals.

- `pd_ipc_conn_interval` = 10
- `pd_ipc_conn_count` = 5

217) pd_ipc_conn_count = *connection-establishment-retry-count*

~<unsigned integer>((0-600))<<50>>

Specifies the number of retries when establishing a connection for sending data between HiRDB servers. Connection establishment retries are made in the following cases:

- A connection timeout has occurred.
- A shortage has occurred in the Listen queue.

This operand is related to the `pd_ipc_conn_interval` operand.

Example

- `pd_ipc_conn_interval` = 10
- `pd_ipc_conn_count` = 5

With these specifications, connection establishment is retried every 10 seconds up to 5 times.

Specification guidelines

Normally, there is no need to specify this operand.

Note

If connection establishing processing is performed in non-block mode (if `Y` is specified for the `pd_ipc_conn_nblock` operand), this operand is ignored.

218) pd_inet_unix_bufmode = os | <u>conf</u>

Specifies the size of the send/receive buffer to be used for communication with HiRDB.

`os`: Use the value set by the OS.

If a send/receive buffer size is specified in any of the following operands, that value is ignored and the value set by the OS is used:

- `pd_tcp_inet_bufsize` operand

- `pd_ipc_inet_bufsize` operand

`conf`: Use the value specified by the following operands:

- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand

Specification guidelines

- Normally, specify `os` in this operand to use the value set by the OS.
- There are advantabes and disadvantages of specifying `conf`, as described below. If you specify `conf`, perform a thorough evaluation including a verification of the machine's performance capabilities.

  ● If the send/receive buffer size is small and large data such as `BLOB` data is sent and received, data resend operations occur. This might adversely affect the communication performance and CPU usage rate. In this case, you can improve the communication performance and reduce the CPU usage rate by increasing the send/receive buffer size.

  ● If the send/receive buffer size is large and small data, such as numeric data, is sent and received, `ACK` might not be returned quickly (delayed `ACK`) when data is received depending on how TCP/IP is configured. This might result in poor performance.

Notes

The send/receive buffer size, OS settings, how to change the size, and the supported size range depend on the OS. For details, see the OS documentation.

Relationship to other operands

This operand is related to the following operands:

- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

The table below lists the send/receive buffer sizes that depend on the combination of the value specified for this operand and the values of the operands related to this operand.

■ Communications in the TCP INET domain

Table 2–3: Communication with clients in the TCP INET domain

No.	Value of the pd_inet_unix_bufmode operand	Values of related operands				Send/receive buffer size that is set
		pd_tcp_inet_bufsize	pd_inet_unix_utl_bufmode	pd_listen_socket_bufset	pd_ipc_inet_bufsize	
1	`os`	--	--	--	--	• `connect`, `accept`, and `listen` sockets: OS setting
2	`conf`	`0`		`0`	`0`	• `connect`, `accept`, and `listen` sockets: OS setting
3					Other than `0`	• `connect` and `listen` sockets: OS setting • `accept` socket: Value of the `pd_ipc_inet_bufsize` operand
4				`1`	`0`	• `connect`, `accept`, and `listen` sockets: OS setting
5					Other than `0`	• `connect` socket: OS setting

No.	Value of the pd_inet_unix _bufmode operand	Values of related operands				Send/receive buffer size that is set
		pd_tcp_inet _bufsize	pd_inet_unix _utl_bufmode	pd_listen_so cket_bufset	pd_ipc_inet _bufsize	
						• `accept` and `listen` sockets: Value of the `pd_ipc_inet_bufsize` operand
6		Other than `0`		`0`	`0`	• `connect` socket: Value of the `pd_tcp_inet_bufsize` operand • `accept` and `listen` sockets: OS setting
7					Other than `0`	• `connect` socket: Value of the `pd_tcp_inet_bufsize` operand • `accept` socket: Value of the `pd_ipc_inet_bufsize` operand • `listen` socket: OS setting
8				`1`	`0`	• `connect` socket: Value of the `pd_tcp_inet_bufsize` operand • `accept` and `listen` sockets: OS setting
9					Other than `0`	• `connect` socket: Value of the `pd_tcp_inet_bufsize` operand • `accept` and `listen` sockets: Value of the `pd_ipc_inet_bufsize` operand

Legend:

--: Ignored if specified.

Table 2–4: Communication with servers in the TCP INET domain (communication without using the pd_utl_buff_size operand)

No.	Value of the pd_inet_unix _bufmode operand	Values of related operands				Send/receive buffer size that is used
		pd_tcp_inet _bufsize	pd_inet_unix _utl_bufmode	pd_listen_so cket_bufset	pd_ipc_inet _bufsize	
1	`os`	--	--	--	--	• `connect`, `accept`, and `listen` sockets: OS setting
2	`conf`			`0`	`0`	• `connect`, `accept`, and `listen` sockets: OS setting
3					Other than `0`	• `connect` and `accept` sockets: Value of the `pd_ipc_inet_bufsize` operand • `listen` socket: OS setting

No.	Value of the pd_inet_unix _bufmode operand	Values of related operands				Send/receive buffer size that is used
		pd_tcp_inet _bufsize	pd_inet_unix _utl_bufmode	pd_listen_so cket_bufset	pd_ipc_inet _bufsize	
4				1	0	• connect, accept, and listen sockets: OS setting
5					Other than 0	• connect, accept, and listen sockets: Value of the pd_ipc_inet_bufsize operand

Legend:

--: Ignored if specified.

Table 2–5: Communication with servers in the TCP INET domain (communication using the pd_utl_buff_size operand)

No.	Value of the pd_inet_unix _bufmode operand	Values of related operands				Send/receive buffer size that is used
		pd_tcp_inet _bufsize	pd_inet_unix _utl_bufmode	pd_listen_so cket_bufset	pd_ipc_inet _bufsize	
1	os	--	--	--	--	• connect, accept, and listen sockets: OS setting
2	conf		conf	0	0	• connect, accept, and listen sockets: OS setting
3					Other than 0	• connect and accept sockets: Value of the pd_ipc_inet_bufsize operand • listen socket: OS setting
4				1	0	• connect, accept, and listen sockets: OS setting
5					Other than 0	• connect, accept, and listen sockets: Value of the pd_ipc_inet_bufsize operand
6			auto	0	0	• connect and accept sockets: Value automatically calculated from the pd_utl_buff_size operand value • listen socket: OS setting
7					Other than 0	• connect and accept sockets: Value automatically calculated from the pd_utl_buff_size operand value • listen socket: OS setting
8				1	0	• connect and accept sockets:

No.	Value of the pd_inet_unix _bufmode operand	Values of related operands				Send/receive buffer size that is used
		pd_tcp_inet _bufsize	pd_inet_unix _utl_bufmode	pd_listen_so cket_bufset	pd_ipc_inet _bufsize	
						Value automatically calculated from the `pd_utl_buff_size` operand value • `listen` socket: OS setting
9					Other than `0`	• `connect` and `accept` sockets: Value automatically calculated from the `pd_utl_buff_size` operand value • `listen` socket: Value of the `pd_ipc_inet_bufsize` operand value

Legend:

--: Ignored if specified.

219) pd_ipc_inet_bufsize = *send-receive-buffer-size-for-server-to-server-communication*

~<unsigned integer>((0-262144)) <<16384>> (bytes)

Specifies, as a multiple of 4,096 bytes, the maximum size of the send/receive buffer to be used for server-to-server communication (TCP INET domain).

Specification guidelines

- Normally, there is no need to specify this operand. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- If you specify `conf` in for the `pd_inet_unix_bufmode` operand, use this operand to specify the send/receive buffer size used for HiRDB communication processing.

Notes

- If `0` is specified, the value set by the OS is used.
- The maximum TCP buffer size depends on the OS.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_tcp_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

220) pd_tcp_inet_bufsize = *send-receive-buffer-size-for-communication-with-HiRDB-client*

~<unsigned integer>((0-262144))<<0>>(bytes)

Specifies as a multiple of 4,096 bytes the maximum size of the send/receive buffer to be used for communication (TCP INET domain) with HiRDB clients.

Specification guidelines

- Normally, this operand does not need to be specified. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- If you specify `conf` in for the `pd_inet_unix_bufmode` operand, use this operand to specify the send/receive buffer size used for HiRDB communication processing.

Notes

- If `0` is specified, the value set by the OS is used.

- The maximum TCP buffer size depends on the OS.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_ipc_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

221) pd_listen_socket_bufset = 0 | 1

Specifies whether the send/receive buffer is to be set in a `listen` socket when the `listen` socket for a TCP INET domain that is used for HiRDB communication processing is created.

`0`: Do not set the send/receive buffer size in the `listen` socket.

`1`: Set the send/receive buffer size in the `listen` socket.

For the send/receive buffer size, the value of the `pd_ipc_inet_bufsize` operand is set. However, the following conditions must be satisfied:

- `conf` is specified for the `pd_inet_unix_bufmode` operand.
- A non-zero value is specified for the `pd_ipc_inet_bufsize` operand.

For details, see the description of the *pd_inet_unix_bufmode* operand.

Specification guidelines

- Normally, this operand does not need to be specified. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- When `conf` is specified for the `pd_inet_unix_bufmode` operand, and the send/receive buffer size specified for this operand is used, if the value of the `pd_ipc_inet_bufsize` operand is smaller than the value set by the OS, a communication delay might occur. In this case, you might be able to improve the communication performance by specifying `1` for this operand.

Notes

If you have specified `1` for this operand, the send/receive buffer size set by this operand is used, even if you specify `conf` for the `pd_utl_inet_unix_bufmode` operand and perform communication using the `pd_utl_buff_size` operand. Set the send/receive buffer size in such a manner that the same size is used for the `listen` and `accept` sockets. If different send/receive buffer sizes are used for the `listen` and `accept` sockets, the communication performance might not be improved.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

222) pd_utl_buff_size = *utility-communication-buffer-size*

~<unsigned integer>((8-64))<<32>>(kilobytes)

Specifies in kilobytes the size of each sector of the buffers used for communication by the following utility processes:

- Database load utility (`pdload`)
- Database reorganization utility (`pdrorg` with `-g` option specified)
- Rebalancing utility (`pdrbal`)
- Database recovery utility (`pdrstr`)
- Database copy utility (`pdcopy`)

Advantage

When data is re-sent in communications between a server process that is processing a utility and a HiRDB server, the number of communication instances can be reduced by increasing the specification of this operand, which can improve performance.

Specification guidelines

$\textit{communication-buffer-size} = \uparrow (\textit{page-size}^{\#} \times 2) \div 1{,}024 \uparrow$ (kilobytes)

#: Maximum page size of RDAREAs to be processed by the utility.

Operand rules

- Specify a multiple of 4. If the specified value is not a multiple of 4, it is rounded up automatically to a multiple of 4.
- If the specified value is smaller than 32, it will be increased automatically to 32 when the database recovery utility or database copy utility is executed.

Relationship to other operands

This operand is related to the operands listed below. For details, see the description of the *pd_inet_unix_bufmode* operand.

- `pd_inet_unix_bufmode` operand
- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_inet_unix_utl_bufmode` operand

Effects on individual estimation formulas

If the value of the `pd_utl_buff_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Memory size required for the execution of the database recovery utility (pdrstr)*
- *Memory size required for the execution of the database load utility (pdload)*
- *Memory size required for the execution of the database copy utility (pdcopy)*

223) pd_inet_unix_utl_bufmode = <u>auto</u> | conf

Specifies the size of the send/receive buffer to be used for communications that use the `pd_utl_buff_size` operand.

`auto`:

Automatically calculate the value based on the `pd_utl_buff_size` operand.

`conf`:

Use the value specified for the `pd_ipc_inet_bufsize` operand.

Specification guidelines

- Normally, this operand does not need to be specified. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- If you specify `conf` for the `pd_inet_unix_bufmode` operand and `1` for the `pd_listen_socket_bufset` operand, specify `conf` for this operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_utl_buff_size` operand

224) pd_utl_file_buff_size = *utility-file-buffer-size*

~<unsigned integer>((32-512000))<<32>>(kilobytes)

Specifies the size of the buffer to be used for I/O when a utility accesses a file. Specifying a large value in this operand can reduce the number of file I/Os.

When inputting and outputting data for files of several hundred gigabytes, the number of I/Os and the required I/O time will likely increase, depending on the OS being used and the hardware characteristics. This can be avoided by specifying a buffer size of around 1 megabyte.

The table below lists the utilities (and their files) for which the buffer size specification in this operand is applicable.

Utility	File
Database copying utility	Backup file
Database recovery utility	Backup file
Database organization utility	Unload data file[#1] LOB data unload file Table transfer unload file
Database creation utility	Input data file[#2] Column-unit LOB input file Table transfer unload file

#1

The buffer size is not applicable for a multi-volume file with the `-W` option specified.

#2

If the `maxreclen` operand was specified for an input data file in DAT format, the `maxreclen` operand specification takes priority. In such a case, increase the value of the `maxreclen` operand specification.

Effects on individual estimation formulas

If the value of the `pd_utl_file_buff_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Memory size required for the execution of the database copy utility (pdcopy)*
- *Memory size required for the execution of the database recovery utility (pdrstr)*
- *Memory size required for the execution of the database load utility (pdload)*

225) pd_sql_send_buff_size = *buffer-size-for-inter-server-communication-for-SQL-execution*

~<unsigned integer>((4-32))<<4>>(kilobytes)

This operand is applicable only to HiRDB/Parallel Server.

Specify the size of each communication buffer (in kilobytes) to be used for sending search result data between HiRDB servers.

Specification guidelines

communication-buffer-size = $\uparrow$ *page-size*[#] $\div$ 1,024 $\uparrow$ (in kilobytes)

#: *page-size* = *row-size* $\times$ *communication-count-per-case*

- For the data size of variable-length data, use the actual size.
- Because LOB columns and abstract data type columns are excluded, do not include them in the page size.

Notes

- Increasing the specification value of this operand might increase the amount of memory used.
- Communication might be conducted using a communication buffer size greater than the specified value in the case of using a feature such as an index, in order to decrease sorting by using a floatable server and for processing to create a work table for that purpose.

2.2.40 Operands related to Java

Java operands are specified when a Java stored procedure or Java stored function is used. For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide*.

226) pd_java_option = "Java-option" [,*"Java-option"*]...

~<character string>

Specifies Java virtual option startup options. For details about the startup options, see the documents on Java virtual machines.

Operand rules

- The maximum character string size of a Java option is 255 characters.
- The total character string size of all Java options is 1,024 characters.
- Double quotation mark (") cannot be specified within a Java option.
- A maximum of 20 Java options can be specified.

Operand specification example

The following example specifies the initial value of the heap size to be used by a Java virtual machine as 32 megabytes and the maximum value of the heap size as 64 megabytes:

```
pd_java_option = "-Xms32m","-Xmx64m"
```

227) pd_java_routine_stack_size = >*stack-size-for-use-by-external-Java-routine*

~<unsigned integer>((1024-2146435072))<<131072>>(bytes)

Specifies in bytes the stack size to be used by an external Java routine.

Specification guidelines

- Specify a value that is greater than the stack size specified as a Java option.
- If an operation for specifying both the stack size and native method stack area size for a Java virtual machine is available as a startup option of the Java virtual machine, specify for the `pd_java_routine_stack_size` operand a value that is greater than the native method stack size.

Notes

- If a value that is smaller than the stack size specified as a Java option is specified, HiRDB might not run correctly.
- If the stack size being used exceeds the value specified by this operand, HiRDB might not run correctly. However, HiRDB will run correctly if the Java virtual machine detects the stack overflow.

228) pd_java_archive_directory = *"JAR-file-storage-directory"*

~<path name> <<%PDDIR%\java>>

Specifies the name of the directory for storing JAR files used by Java stored procedures or Java stored functions.

Notes

- The specified directory must be created before installation of JAR files.
- The JAR file storage directory is used only for storing JAR files.
- Store only the installed JAR files in the JAR file storage directory.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

229) pd_java_classpath = *"Java-class-path"*

~<path name>

Specifies as an absolute path name the class path to be used by a Java virtual machine.

The class contained in the path specified by this operand can be referenced from the Java method, which is executed as the processing procedure of a Java stored procedure or Java stored function.

If a class with the same name exists in the path specified by this operand and in the JAR file specified as an external routine of the Java stored procedure or Java stored function, the path specified by this operand takes precedence.

Operand rules

- Up to 1,024 characters can be used for the path name.
- Path names are not case sensitive.

230) pd_java_runtimepath = "*Java-Runtime-Environment-root-directory*"

~<path name> <<%PDDIR%\jre>>

Specifies as an absolute path name the root directory of the Java Runtime Environment.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

Notes

Because Java Runtime Environment (JRE) is no longer included, you must add this operand or change its value when you upgrade HiRDB from a version earlier than 07-03 to version 07-03 or later. For notes about upgrading, see *Using Java stored procedures and functions* in the *HiRDB Version 9 Installation and Design Guide*.

231) pd_java_libpath = "*Java-virtual-machine-library-directory*"

~<path name>

Specifies the directory that stores the library of the Java virtual machine as a relative path name to the Java Runtime Environment root directory (the value of the `pd_java_runtimepath` operand).

Operand default values

The following table lists the default values for this operand, depending on the OS:

OS	Operand's default value
Windows (excluding the following)	`bin\hotspot`, or `bin\client`#
Windows Server 2003 (IPF)	`bin\server`
Windows (x64)	

#

Note that `bin\client` is assumed if `jvm.dll` does not exist in `bin\hotspot`.

Operand rules

- A maximum of 255 characters can be used for the path name.
- Path names are not case sensitive.

232) pd_java_stdout_file = "*Java-virtual-machine-standard-output-and-standard-error-output-destination-file*"

~<path name>

Specifies as an absolute path name the file to which the standard output and standard error output are to be output in a Java virtual machine. If this operand is omitted, the standard output and standard error output of the Java virtual machine are ignored.

Specification guidelines

Because the size of the file specified by this operand is extremely large, this operand is not normally specified. It is recommended that this operand be specified during debugging of a Java stored procedure or Java stored function. There is no limit to the size of the file that can be specified by this operand.

Note

If there are simultaneous writing attempts from multiple processes, their output contents cannot be guaranteed.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

2.2.41 Operands related to external C stored routines

233) pd_c_library_directory = "*C-library-file-storage-directory*"
~<path name><<%PDDIR%\clib>>

Specifies the absolute path name of the directory that stores C library files.

Operand rules

- A maximum of 255 characters can be used for the path name.
- Path names are not case sensitive.

Note

When this operand is used, the directory specified here must be created before C library files can be registered.

2.2.42 Operands related to character encoding

234) pd_substr_length = <u>3</u> | 4 | 5 | 6

Specifies the maximum number of bytes to be used to represent a single character. This operand is applicable when Unicode (UTF-8) is specified as the character encoding and affects the length of the results of the `SUBSTR` scalar function.

For details about the `SUBSTR` scalar function, see the manual *HiRDB Version 9 SQL Reference*.

Specification guidelines

If you use only characters in the UCS-2 range (1 to 3 bytes), you can omit this operand. To use characters in the UCS-4 range (1 to 6 bytes), you must specify in this operand the maximum number of bytes to be used to represent a single character.

If the specified value is greater than the actual number of bytes used to represent a character, the result of the `SUBSTR` scalar function might become too long. For this reason, evaluate the value to be specified in this operand on the basis of the character encoding actually used.

Notes

- According to ISO/IEC Standard 10646, a range of one to four bytes is assigned per character and bytes 5 and 6 are reserved for future standards. Although HiRDB supports a range of one to six bytes per character, problems might arise in the future if you use a range of 5 or 6 bytes but no characters have been assigned using that number of bytes.
- In Unicode (UTF-8) supported by version 06-02, you can use a maximum of three bytes per character. Version 08-00 supports a maximum of six bytes. Therefore, with some data, the number of bytes representing a single character might increase.
- If all the following conditions are satisfied, you must re-create SQL objects in routines and re-define view tables:

  ● Unicode (UTF-8) is specified as the character encoding.

  ● A value other than `3` is to be specified in the new `pd_substr_length` operand; or, in an environment where the `pd_substr_length` operand has already been specified, routines and view tables are to be defined and the `pd_substr_length` operand value is to be changed.

  ● A `SUBSTR` scalar function with a mixed character string type (`MCHAR` or `MVARCHAR`) specified as its argument is to be specified in routines and view definitions.

  You can specify the `PDSUBSTRLEN` operand in the client environment definition to control the connection, or you can specify the `SUBSTR LENGTH` SQL compile option to control a routine.
- The tables below show the priority among the `pd_substr_length` operand and other operands when Unicode (UTF-8) is specified as the character encoding.

  When control is performed by routine

Control timing	pd_substr_length operand	Client environment definition PDSUBSTRLEN	SQL compile option SUBSTR LENGTH	Remarks
During definition	Valid (2)	Invalid	Valid (1)	Default value `pd_substr_length`
During call or function call	Invalid (depends on the specification made during definition)	Invalid (depends on the specification made during definition)	--	None

Legend:
( ): Priority level is indicated within the parentheses
--: Cannot be specified

When control is performed by view table

Control timing	pd_substr_length operand	Client environment definition PDSUBSTRLEN	SQL compile option SUBSTR LENGTH	Remarks
During definition	Valid (2)	Valid (1)	--	Default value `pd_substr_length`
During operation	Invalid (depends on the specification made during definition)	Invalid (depends on the specification made during definition)	--	None

Legend:
( ): Priority level is indicated within the parentheses
--: Cannot be specified

2.2.43 Operands related to date and time

235) pd_leap_second = Y | N

Specifies whether leap seconds are to be applied in time data, timestamp data, and date/time formats.

`Y`:

- For time data or timestamp data
  Applies leap seconds in embedded variables and `?` parameter values, as well as in values that represent them in character strings.
- For date/time formats
  Applies leap seconds in values for seconds.

`N`:

Does not apply leap seconds.

Note

When time data or timestamp data that includes leap seconds is stored in a database, specifying `N` in this operand might result in an error when time operations are performed using time data or timestamp data that includes leap seconds. If this occurs, you can use the `SECOND` scalar function to extract the leap seconds.

2.2.44 Operands related to unit structure

236) pdunit -x *host-name*
-u *unit-identifier*
[-d *"HiRDB-directory-name"*]
[-c *host-name*]
[-p *HiRDB-port-number*]
[-s *scheduler-process-port-number*]
[-t *transaction-server-process-port-number*]
[-m *message-log-server-process-port-number*]
[-a *unit-monitoring-process-port-number*]

Remarks

You can change the `-x` and `-c` option values when HiRDB is restarted following a planned termination. For the other options, change their values after HiRDB is started normally.

HiRDB/Single Server

Defines the unit structure of a HiRDB/Single Server.

Specification guidelines

- You can omit this operand for a HiRDB/Single Server.
  However, it cannot be omitted if a system switchover facility that does not inherit IP addresses is used.
- When a system switchover facility is used, the definitions must be the same for the primary system and the secondary system.
- For a mutual system switchover configuration, in which a multi-HiRDB configuration is used (two HiRDB/Single Servers coexisting in a single server machine), the same system definition cannot be used for both HiRDB/Single Servers.
- Specify different names for the hosts specified in the `-x` and `-c` options of all `pdunit` operands that are specified.

-x *host-name*

~<host name>((1-32 characters))

Specifies the host name of the server machine that defines the HiRDB/Single Server.

This operand can be specified as an IP address or in FQDN format. A loopback address can also be specified. However, the same specification format must be used consistently within a server machine. For example, once a host is specified in host name format, a host must not be specified as an IP address on the same server machine.

For specification examples, see *Settings for a DNS server to manage IP addresses* in the *HiRDB Version 9 Installation and Design Guide*.

Caution about specifying in host name format

- Specify the host name displayed when the `hostname` command is executed.
- Host names are case sensitive.
- You cannot specify an alias for a host name.

Relationship to other operands

The following table shows the relationships between the host names specified in the `-x` and `-c` options of the `pdunit` operand and in the `pd_hostname` operand:

System configuration condition	pdunit operand		pd_hostname operand specification
	-x option specification	-c option specification	
Not using switchover facility	• Specify host name#1 • Omit `pdunit` operand	Need not be specified	• Omit `pd_hostname` operand • Specify standard host name#1 • Specify same host name as in `-x` option#1

System configuration condition		pdunit operand		pd_hostname operand specification
		-x option specification	-c option specification	
		Specify loopback address [#2, #4]	Need not be specified	• Omit `pd_hostname` operand • Specify loopback address
Using switchover facility	Monitor mode (IP addresses inherited)[#6]	Specify relocatable IP address, host name with a relocatable IP address, or FQDN	Need not be specified	• Omit `pd_hostname` operand • Specify primary system's standard host name[#1] • Specify same host name as in `-x` option[#1]
		Specify loopback address[#2, #3, #4, #5]	Need not be specified	• Omit `pd_hostname` operand • Specify loopback address
	IP addresses not inherited	Specify primary system's host name[#1]	Specify secondary system's host name[#1]	Specify primary system's standard host name[#1]

Notes:

- A host name specified here must be registered in the `hosts` file, DNS, or similar, and the name must be resolved.
- If a host is specified in FQDN format, it must have been defined in FQDN format.
- In HiRDB, `localhost` is normally treated as a host name, so when used as a loopback address, the name must be resolved.
- When the `pdunit` operand is omitted, it is assumed that the standard host name has been specified with the `-x` option. In such a case, a standard host name must be applied following the rules described in *Syntax element conventions* in the *Preface*.
- When the `pdunit` operand is omitted, the standard host name is assumed, so a standard host name must be registered in the `hosts` file, DNS, or similar, and the name must be resolved.

#1

The specification format can be a host name, IP address, or FQDN format.

#2

If a loopback address is specified, it is not necessary to register the host name in the `hosts` file, DNS, or similar.

#3

In this case, it is not necessary to set up cluster software IP address inheritance.

#4

In this case, HiRDB clients on other machines cannot connect to the HiRDB server. To connect to the HiRDB server, use the multi-connection address facility or the high-speed connection facility.

#5

When using the multi-connection address facility, specify in the `-m` option of the `pdstart` operand (the option that specifies the host name to which the HiRDB client connects) a host name with a relocatable IP address that can be inherited at switchover.

#6

Specify the virtual network name to be registered in the MSCS or MSFC resource (network name). For details about virtual network name registration, see the *HiRDB Version 9 System Operation Guide*.

-u *unit-identifier*

~<identifier>((4 characters))

Specifies a unit identifier. The unit identifier specified here is also specified in the `pd_unit_id` operand in the unit control information definition.

-d "*HiRDB-directory-name*"

~<path name of up to 200 characters><<value of environment variable PDDIR>>

Specifies the name of the HiRDB directory.

The HiRDB directory name is not case sensitive. `C:\hirdb_x` is treated as the same as `C:\HIRDB_X`.

When the `pdunit` operand is omitted, the installation directory is assumed.

-c *host-name*

~<host name>((1-32 characters))

Specifies the secondary system's host name in one of the formats listed below. You specify this option for system switchover in which IP addresses are not inherited.

Use this option to specify the secondary system's host name. This option can be specified as an IP address or in FQDN format. However, the same specification format must be used consistently within a server machine. For example, once a host is specified in host name format, a host must not be specified as an IP address on the same server machine.

Caution about specifying in host name format

- Host names are case sensitive.
- You cannot specify an alias for a host name.

-p *HiRDB-port-number*

~<unsigned integer>((5001-65535))

Do not specify this option for a HiRDB/Single Server.

-s *scheduler-process-port-number*

~<unsigned integer>((5001-65535))

Specifies the port number for the scheduler process.

Notes

- Normally, this operand need not be specified.
- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- For details about how to specify a scheduler process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand is related to the following operands:

- `pd_service_port`
- `pd_scd_port`

-t *transaction-server-process-port-number*

~<unsigned integer>((5001-65535))

Specifies the port number for the transaction server process.

Notes

- Normally, this operand need not be specified.
- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- For details about how to specify a transaction server process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand is related to the `pd_trn_port` operand.

-m *message-log-server-process-port-number*

~<unsigned integer>((5001-65535))

Specifies the port number of the message log server process.

Notes

- Normally, this operand need not be specified.
- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- For details about how to specify a message log server process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand is related to the `pd_mlg_port` operand.

HiRDB/Parallel Server

Defines the unit structure of a HiRDB/Parallel Server.

Specification guidelines

- Information on all units that comprise the HiRDB/Parallel Server must be defined with this operand. For example, if there are three units, three `pdunit` operands are required.
- This operand cannot be omitted when a system switchover facility is used.
- When a system switchover facility is used, the definitions must be the same for the primary system and the secondary system.
- For a mutual system switchover configuration, in which a multi-HiRDB configuration is used (two units coexisting in a single host), the same definition cannot be used for the two separate units.
- Specify different names for the hosts specified in the `-x` and `-c` options of all `pdunit` operands that are specified.
- If the HiRDB/Parallel Server consists of a single unit, this operand can be omitted. In this case, the installation directory is assumed as the HiRDB directory.

-x *host-name*

~<host name>((1-32 characters))

Specifies the host name of the server machine that defines the server of the HiRDB/Parallel Server. This operand can be specified as an IP address or in FQDN format. However, the same specification format must be used consistently within a server machine. For example, once a host is specified in host name format, a host must not be specified as an IP address on the same server machine.

For specification examples, see *Settings for a DNS server to manage IP addresses* in the *HiRDB Version 9 Installation and Design Guide*.

Caution about specifying in host name format

- Specify the host name displayed when the `hostname` command is executed.
- Host names are case sensitive.
- You cannot specify an alias for a host name.

Relationship to other operands

- The following table shows the relationships between the host names specified in the `-x` and `-c` options of the `pdunit` operand and in the `pd_hostname` operand:

System configuration condition		pdunit operand		pd_hostname operand specification
		-x option specification	-c option specification	
Not using switchover facility		Specify host name.#1	Does not need to be specified.	• Omit `pd_hostname` operand. • Specify standard host name.#1
Using switchover facility	IP addresses inherited#3	Specify relocatable IP address, host name with a relocatable IP address, or an FQDN.	Does not need to be specified.	Specify primary system's standard host name.#1
	IP addresses not inherited	Specify primary system's host name.#1	Specify secondary system's host name.#1, #2	Specify primary system's standard host name.#1

Note:

- A host name specified here must be registered in the `hosts` file, DNS, or similar, and the name must be resolved.
- If a host is specified in FQDN format, it must have been defined in FQDN format.

#1

The specification format can be a host name, IP address, or FQDN format.

#2

The same host name cannot be specified more than once in the `-x` and `-c` options of the `pdunit` operand. The following shows examples of valid and invalid specifications:

Valid examples

```
pdunit -x hostA ... -c hostAA
pdunit -x hostB ... -c hostBB
```

Invalid examples

```
pdunit -x hostA ... -c hostB
pdunit -x hostB ... -c hostA
```

For an example of a HiRDB system definition that uses system switchover and in which IP addresses are not inherited, see *B.4 HiRDB/Parallel Server: when the standby system switchover facility is used*.

#3

Specify the virtual network name to be registered in the MSCS or MSFC resource (network name). For details about virtual network name registration, see the *HiRDB Version 9 System Operation Guide*.

-u *unit-identifier*

~<identifier>((4 characters))

Specifies a unit identifier. The unit identifier specified here is also specified in the `pd_unit_id` operand in the unit control information definition of each unit.

-d "*HiRDB-directory-name*"

~<path name of up to 200 characters><<value of environment variable PDDIR>>

Specifies the name of the HiRDB directory for this unit.

The HiRDB directory name is not case sensitive. `C:\hirdb_x` is treated as the same as `C:\HIRDB_X`.

Notes

- When using the standby-less system switchover (1:1) facility
  Specify the same HiRDB directory name in the normal and back-end server units.
- When using the standby-less system switchover (effects distributed) facility
  Specify the same HiRDB directory name in all units belonging to the same HA group.
- The environment variable `PDDIR` is set to the environment variable that is set in the unit that executed the `pdstart -q` command. If you omit the `-d` option for a system composed of multiple units, specify the same HiRDB directory name for all units.

-c *host-name*

~<host name>((1-32 characters))

Specifies the secondary system's host name in one of the formats listed below. You specify this option for system switchover in which IP addresses are not inherited.

Use this option to specify the secondary system's host name. This option can be specified as an IP address or in FQDN format. However, the same specification format must be used consistently within a server machine. For example, once a host is specified in host name format, a host must not be specified as an IP address on the same server machine.

This option cannot be specified for the units described below. If it is specified, HiRDB cannot start (message `KFPS01896-E` is output).

- Unit to which the standby-less system switchover (1:1) facility is applied. (IP address inheriting does not occur.)
- Unit to which the standby-less system switchover (effects distributed) facility is applied. (IP address inheriting does not occur.)

Caution about specifying in host name format

- Host names are case sensitive.
- You cannot specify an alias for a host name.

-p *HiRDB-port-number*

~<unsigned integer>((5001-65535))

Specifies the HiRDB port number for the unit. When this option is omitted, the value specified in the `pd_name_port` operand is assumed.

Notes

- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- When using the standby-less system switchover (1:1) facility, specify the same port number for the normal BES unit and the alternate BES unit.
- When using the standby-less system switchover (effects distributed) facility, specify the same port number for all units belonging to the same HA group.
- For details about how to specify a HiRDB port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

This operand is related to the `pd_name_port` operand.

-s *scheduler-process-port-number*

~<unsigned integer>((5001-65535))

When `Y` is specified in the `pd_name_fixed_port_lookup` operand, specifies the port number for the scheduler process for the unit.

Notes

- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- When using the standby-less system switchover (1:1) facility, specify the same port number for the normal BES unit and the alternate BES unit.
- When using the standby-less system switchover (effects distributed) facility, specify the same port number for all units belonging to the same HA group.
- For details about how to specify a scheduler process port name and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.
- This operand is related to the following operands:
  - `pd_service_port`
  - `pd_scd_port`

-t *transaction-server-process-port-number*

~<unsigned integer>((5001-65535))

When `Y` is specified in the `pd_name_fixed_port_lookup` operand, specifies the port number for the transaction server process for the unit.

Notes

- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- When using the standby-less system switchover (1:1) facility, specify the same port number for the normal BES unit and the alternate BES unit.
- When using the standby-less system switchover (effects distributed) facility, specify the same port number for all units belonging to the same HA group.
- For details about how to specify a transaction server process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.
- This operand is related to the `pd_trn_port` operand.

-m *message-log-server-process-port-number*

~<unsigned integer>((5001-65535))

When `Y` is specified in the `pd_name_fixed_port_lookup` operand, specifies the port number of the message log server process for the unit.

Notes

- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- When using the standby-less system switchover (1:1) facility, specify the same port number for the normal BES unit and the alternate BES unit.
- When using the standby-less system switchover (effects distributed) facility, specify the same port number for all units belonging to the same HA group.
- In the case of a unit on which no message log server process is running, a port number cannot be used, even if one is specified. However, such a defined port number is checked to ensure it does not duplicate any other port number. A unit on which no message log server process is running is a unit without a system manager when a HiRDB/Parallel Server is configured of at least two units and `manager` is specified in `pd_mlg_msg_log_unit`.
- For details about how to specify a message log server process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.
- This operand is related to the `pd_mlg_port` operand.

-a *unit-monitoring-process-port-number*

~<unsigned integer>((5001-65535))

When `Y` is specified in the `pd_name_fixed_port_lookup` operand, specifies the port number of the unit monitoring process for the unit.

Notes

- When a single server machine has multiple units, as in a mutual switchover configuration, specify a different port number for each unit.
- When using the standby-less system switchover (1:1) facility, specify the same port number for the normal BES unit and the alternate BES unit.
- When using the standby-less system switchover (effects distributed) facility, specify the same port number for all units belonging to the same HA group.
- In the case of a unit on which no unit monitoring process is running, a port number cannot be used, even if one is specified. However, such a defined port number is checked to ensure that it does not duplicate any other port number. A unit monitoring process does not run on any unit other than a unit with a system manager when a HiRDB/Parallel Servers consists of at least two units.
- For details about how to specify a unit monitoring process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

Relationship to other operands

- `Y` must be specified in the `pd_name_fixed_port_lookup` operand.
- This operand is related to the `pd_alv_port` operand.

2.2.45 Operands related to server structure

237) pdstart -t *server-type* [-s server-name]

-x *host-name* | -u *unit-identifier*

[-m *host-name*[,*host-name*]...

[-n *host-name*[,*host-name*]]

[-c *server-name* | -g *HA-group-identifier*]

[-k stls]

This operand must be specified; it cannot be omitted.

Notes

- To change a server name that has been specified (with the `-s` option specification), you need to use the database initialization utility to reconstruct the system. Therefore, avoid server names that might have to be changed later.

- To change the specification of the `-t`, `-u`, `-s`, or `-c` option, you need to perform the operations described below. Therefore, avoid specifications that might have to be changed later.

1. Initialization of the system log file (The system log file of the server for which the option was changed must be initialized.)
2. Initialization of the synchronization point dump file (The synchronization point dump file of the server for which the option was changed must be initialized.)
3. Initialization of the status file for unit and the status file for server (The status files for the unit and server for which the options were changed must be initialized.)

Remarks

You can change the `-x`, `-m`, and `-n` option values when HiRDB is restarted following a planned termination. For the other options, change their values after HiRDB is started normally.

HiRDB/Single Server

The server configuration of a HiRDB/Single Server is defined. The following items can be specified:

- Server type
- Server name
- Host name or unit identifier

Specification guideline

If multiple HiRDB/Single Servers are connected and used, a unique server name must be specified for each HiRDB/Single Server.

-t *server-type*

Specifies the server type:

`SDS`: Single server

-s *server-name*

~<identifier>((1-8 characters))

Specifies the server name of the single server.

The server name is not case sensitive. `sds01` and `SDS01` are treated as the same.

-x *host-name*

~<host name>((1-32 characters))

Specifies the host name specified for the `-x` option of the `pdunit` operand.

If the `pdunit` operand is omitted, HiRDB assumes that the standard host name is specified in the `-x` option of the `pdunit` operand. Therefore, specify the standard host name in this option.

-u *unit-identifier*

~<identifier>((4 characters))

Specifies the identifier of the unit that executes the server. Specifies the unit identifier specified by the `-u` option of the pdunit operand.

When the system switchover facility is used, specification of `-u unit-identifier` instead of `-x host-name` is recommended.

-m *host-name*[,*host-name*]...

~<host name>((1-32 characters))

Specifies the host name of the HiRDB/Single Server to which the HiRDB client connects using the *multi-connection address facility*. You can also specify this value in the IP address or FQDN format.

For details about the multi-connection address facility, see the *HiRDB Version 9 Installation and Design Guide*.

Specify this option when the network being used between HiRDB clients and HiRDB servers is different from the network being used between the server machines of the HiRDB servers. If a loopback address is specified in the `pd_hostname` operand, this option must be specified in order to access the HiRDB server from another host's HiRDB client.

Caution about specifying in host name format

- Host names are case sensitive.
- You cannot specify aliases for host names.

Notes

- You can specify a maximum of four this option.
- If you specify multiple host names that are on the same network, the first host name specified is valid.
- If you specify the host name of a server machine not containing a HiRDB/Single Server, that specification is ignored.
- If the network for connecting HiRDB clients is separated from the network for communicating between HiRDB servers by a subnet, the multi-connection address facility cannot be used. If the `-m` option is specified, the HiRDB client might not be able to connect to the HiRDB server.
- When you are using a system switchover facility that does not inherit IP addresses, also specify the `-n` option. Specify the host name of the primary system for the `-m` option, and specify the host name of the secondary system for the `-n` option.
- If the system switchover facility that inherits IP addresses is used and a loopback address is specified in the `-x` option of the `pdunit` operand, specify the host name with a relocatable IP address in the `-m` option.

-n *host-name*[,*host-name*]...

~<host name>((1-32 characters))

Specify this option together with the `-m` option.

Specifies the host name of the HiRDB/Single Server to which the HiRDB client connects using the multi-connection address facility. You can specify this value also in the IP address or FQDN format.

Specify this option when the network being used between HiRDB clients and HiRDB servers is different from the network being used between the server machines of the HiRDB servers. When you are using a system switchover facility that does not inherit IP addresses, specify the host name of the secondary system.

Caution about specifying in host name format

- Host names are case sensitive.
- You cannot specify aliases for host names.

Notes

- You can specify a maximum of four this option.
- If you specify multiple host names that are on the same network, the first host name specified is valid.
- If you specify the host name of a server machine not containing a HiRDB/Single Server, that specification is ignored.
- If the network for connecting HiRDB clients is separated from the network for communicating between HiRDB servers by a subnet, the multi-connection address facility cannot be used. If the `-n` option is specified, the HiRDB client might not be able to connect to the HiRDB server.
- If you specify this option, specify the standard host name in the `pd_hostname` operand.

-c *server-name*

~<identifier>((1-8 characters))

Omit this option, because it is used only for a HiRDB/Parallel Server.

-g *HA-group-identifier*

~<identifier>((1-8 characters))

Omit this option because it applies only to a HiRDB/Parallel Server.

-k stls

~<identifier>((4 characters))

Omit this option because it applies only to a HiRDB/Parallel Server.

HiRDB/Parallel Server

The server configuration of a HiRDB/Parallel Server is defined. The following items can be specified:

- Server type
- Server name
- Host name or unit identifier

Specification guidelines

The following guidelines are for the server configuration:

- Concurrent execution of SQL statements can be improved if the individual servers are distributed among separate server machines.
- Concurrent execution of SQL statements can be improved if the system manager and front-end servers are defined in server machines connected using the TCP/IP protocol.
- In some cases, it might be better to allocate multiple back-end servers to a single server machine, depending on the CPU workloads of the back-end servers.

Note

Only one system manager server and one dictionary server can be specified.

-t *server-type*

Specifies the server type:

`MGR`: System manager

`FES`: Front-end server

`BES`: Back-end server

`DIC`: Dictionary server

-s *server-name*

~<identifier>((1-8 characters))

Specifies a server name. This option need not be specified if the server type is MGR (system manager).

The server name is not case sensitive. `bes01` and `BES01` are treated as the same.

-x *host-name*

~<host name>((1-32 characters))

Specifies the host name specified for the `-x` option of the `pdunit` operand.

-u *unit-identifier*

~<identifier>((4 characters))

Specifies the identifier of the unit that executes the server. Specifies the unit identifier specified by the `-u` option of the pdunit operand.

When the system switchover facility is used, specification of `-u unit-identifier` instead of `-x host-name` is recommended.

-m *host-name*[,*host-name*]...

~<host name>((1-32 characters))

Specifies the host name of the front-end server to which the HiRDB client connects using the *multi-connection address facility*. You can specify this value also in the IP address or FQDN format.

For details about the multi-connection address facility, see the *HiRDB Version 9 Installation and Design Guide*.

Specify this option when the network being used between HiRDB clients and HiRDB servers is different from the network being used between the server machines of the HiRDB servers.

Caution about specifying in host name format

- Host names are case sensitive.
- You cannot specify aliases for host names.

Notes

- You can specify a maximum of four this option.
- If you specify multiple host names that are on the same network, the first host name specified is valid.
- If you specify the host name of a server machine not containing a front-end server, that specification is ignored.
- If the network for connecting HiRDB clients is separated from the network for communicating between HiRDB servers by a subnet, the multi-connection address facility cannot be used. If the `-m` option is specified, the HiRDB client might not be able to connect to the HiRDB server.

- When you are using a system switchover facility that does not inherit IP addresses, also specify the `-n` option. Specify the host name of the primary system for the `-m` option, and specify the host name of the secondary system for the `-n` option.
- If the system switchover facility that inherits IP addresses is used and a loopback address is specified in the `-x` option of the `pdunit` operand, specify the host name with a relocatable IP address in the `-m` option.

-n *host-name*[,*host-name*]...

~<host name>((1-32 characters))

Specify this option together with the `-m` option.

Specifies the host name of the front-end server to which the HiRDB client connects using the multi-connection address facility. You can specify this value also in the IP address or FQDN format.

Specify this option when the network being used between HiRDB clients and HiRDB servers is different from the network being used between the server machines of the HiRDB servers. When you are using a system switchover facility that does not inherit IP addresses, specify the host name of the secondary system.

Caution about specifying in host name format

- Host names are case sensitive.
- You cannot specify aliases for host names.

Notes

- You can specify a maximum of four this option.
- If you specify multiple host names that are on the same network, the first host name specified is valid.
- If you specify the host name of a server machine not containing a front-end server, that specification is ignored.
- If the network for connecting HiRDB clients is separated from the network for communicating between HiRDB servers by a subnet, the multi-connection address facility cannot be used. If the `-n` option is specified, the HiRDB client might not be able to connect to the HiRDB server.
- If you specify this option, specify the standard host name in the `pd_hostname` operand.

-c *server-name*

~<identifier>((1-8 characters))

This option is related to the standby-less system switchover (1:1) facility. Specify for this option the name of the alternate BES name of the back-end server specified by the `-s` option.

If you specify the `-c` option, all of the following conditions must be satisfied. Otherwise, HiRDB cannot be started. Here, it is assumed that the `-u` option is specified for the `pdstart` operand.

1. `BES` must be specified for the `-t` option.
2. The normal BES unit and the alternate BES unit must not contain any servers except back-end servers.
3. Specify the alternate BESs for all of the back-end servers in the normal BES unit (the unit specified in the `-u` option). Additionally, you must specify those BESs in the same unit (alternate BES unit).
4. You cannot specify duplicate server names in the `-c` option in the normal BES unit.
5. The number of back-end servers in the normal BES unit must be the same as the number of back-end servers in the alternate BES unit.
6. If you specify the `-c` option for the `pdstart` operand of the alternate BES, you must specify for this option the back-end servers inside the normal BES unit.

-g *HA-group-identifier*

~<identifier>((1-8 characters))

When you use the standby-less system switchover (effects distributed) facility, this option specifies an *HA group* identifier. The HA group identifier is a set of units that becomes the destination to which the server specified by the `-s` option is to be moved, and must be specified in the `pdhagroup` operand. For details about HA groups, see the *HiRDB Version 9 System Operation Guide*.

When you specify this option, all of the following conditions must be satisfied. Otherwise, HiRDB cannot start.

- **Condition for the server to be specified in the -s option**

- `BES` is specified for the `-t` option. (A back-end server must be specified.)
- The -c option is not specified. (An alternate back-end server cannot be specified for this server.)

● **Condition for the unit to be specified in the -u option**

- The normal unit belongs to the HA group specified by the `-g` option.
- The normal unit is comprised of back-end servers only.
- An HA group is specified as the moving destination for the servers comprising the normal unit.

● **Condition for the HA group to be specified in the -g option**

- All units within the HA group belong to the same network segment.
- The standby-less system switchover (effects distributed) facility is being applied. (`pd_ha_agent = activeunits` is specified.)
- At least one host back-end server exists inside the unit.
- The unit is comprised of back-end servers only.
- All servers within the unit belong to an HA group.

-k stls

~<identifier>((4 characters))

Specify this option when using a *recovery-unnecessary front-end server*. To use a recovery-unnecessary front-end server, you need the HiRDB Non Recover FES.

If any of the following conditions is satisfied, HiRDB cannot start.

- Specification for this option contains an error.
- The server specified for the `-t` option is not a front-end server.
- This option is specified for a unit containing a server that is not a front-end server.
- HiRDB Non Recover FES is not set up.

Relationship to other operands

- If you specify this option, re-estimate the value of the `pd_log_max_data_size` operand.
- If this option is specified for a front-end server, that front-end server's unit in the `pd_start_skip_unit` operand is ignored, if specified.
- The system switchover facility is not applicable to a recovery-unnecessary front-end server unit. If your system uses the system switchover facility, make sure that `nouse` is specified in the `pd_ha_unit` operand for the recovery-unnecessary front-end server unit.

2.2.46 Operands related to the global buffers

238) pdbuffer *-a buffer-name*

{-r *RDAREA-name*[,*RDAREA-name*]... |

-b *RDAREA-name*[,*RDAREA-name*]... |

-o |

-i *authorization-identifier.index-identifier* }

-n *buffer-sectors-count* [-l *buffer-size*]

[-m *maximum-concurrently-executable-prefetches-count*]

[-p *maximum-batch-input-pages-count*]

[-w *updated-pages-output-rate-during-deferred-write-trigger*]

[-c]

[-y *update-buffer-sectors-count-for-deferred-write-trigger-event*]

Specifies the RDAREAs to which a global buffer is to be allocated. A global buffer is an area for storing data during input of table data to or output of table data from RDAREAs; such a buffer is allocated in the shared memory.

Global buffers must be allocated to all RDAREAs. An SQL statement or the `pdopen` command cannot be executed for an RDAREA to which a global buffer has not been allocated. For details about how to design global buffers, see the *HiRDB Version 9 Installation and Design Guide*.

Operand rules

You can specify the `pdbuffer` operand up to 2,000,000 times. For a HiRDB/Parallel Server, you can specify this operand up to 2,000,000 times per server. However, the overall limit is 2,147,483,647.

Tuning the specified values

For details about how to tune global buffers, see the *HiRDB Version 9 System Operation Guide.*

Operand default value

If the `pdbuffer` operand is omitted, global buffers are allocated under the following conditions:

- Global buffer name: `gbuf00`
- Global buffer type: As specified by the `-o` option
- Global buffer size: 6 MB
- Number of buffer sectors: 6 MB ÷ *maximum-RDAREA-page-size*

-a *buffer-name* ~<identifier>((1-16 characters))

Specifies a name for the global buffer. The same name cannot be repeated.

-r *RDAREA-name*[,*RDAREA-name*]...

~<identifier>((1-30 characters))

Specifies the names of RDAREAs to which the global buffer is to be allocated. The names of the following types of RDAREAs can be specified:

- Master directory RDAREA
- Data dictionary RDAREAs
- Data directory RDAREA
- User RDAREAs
- Data dictionary LOB RDAREAs#
- User LOB RDAREAs#
- Registry RDAREA
- Registry LOB RDAREA#
- List RDAREA

#: It is recommended that a LOB global buffer defined with the `-b` option be allocated to a LOB RDAREA (by specifying both the `-r` and `-b` options).Allocating LOB global buffers improves performance.

Specification guidelines

When RDAREAs with the same or similar page sizes are allocated to the same global buffer, the number of inputs/outputs can be reduced. However, in the following cases, the number of inputs/outputs can be reduced if RDAREAs with the same page size are allocated to different global buffers:

- RDAREAs that have tables with different purposes
- RDAREAs that involve many random accesses and RDAREAs that involve sequential accesses

Operand rules

- If the name of an RDAREA includes characters other than single-byte alphanumeric characters, the name must be enclosed in quotation marks (").
- When an RDAREA name is enclosed in quotation marks ("), the name becomes case sensitive (lowercase characters are distinguished from uppercase characters). However, if an RDAREA name is not enclosed in quotation marks, all the characters are handled as uppercase characters.
- A maximum of 4,096 RDAREAs can be defined for a single global buffer.

-b *RDAREA-name*[,*RDAREA-name*]...

~<identifier>((1-30 characters))

Specifies the names of RDAREAs to which a LOB global buffer is to be allocated. The names of the following types of RDAREAs can be specified:

- Data dictionary LOB RDAREAs
- User LOB RDAREAs
- Registry LOB RDAREA

A LOB RDAREA specified here must also be specified in the `-r` option. If the `-b` option only is specified, the LOB RDAREA cannot be accessed. A specification example follows:

Examples

Global buffers are allocated to a LOB RDAREA (RDLOB01) by specifying the `-r` and `-b` options:

```
pdbuffer -a gbuf01 -r RDLOB01 -n 1000
pdbuffer -a gbuf02 -b RDLOB01 -n 1000
```

Remarks

- The LOB RDAREA consists of a directory portion and a data portion. These portions are managed by different global buffers. The directory portion is cached in the global buffer with the `-r` option and the data portion is cached in the global buffer with the `-b` option. Therefore, both the `-r` (or `-o`) and `-b` options must be simultaneously specified.
- Because the global buffer with the `-r` option and the global buffer with the `-b` option are used for different purposes, their sizes must be separately estimated. Because the global buffer with the `-r` option caches only the directory portion, it can be relatively small. In contrast, the global buffer with the `-b` option caches the data portion of the LOB RDAREA. Therefore, estimate the size of this global buffer by taking into consideration both the available memory size and hit rates. For details about how to estimate global buffer sizes, see the *HiRDB Version 9 Installation and Design Guide*.

Specification guidelines

The following types of LOB RDAREAs can be specified:

- LOB RDAREAs for storing plug-in indexes
- LOB RDAREAs for storing a small volume of data that is accessed frequently

It is recommended that a single LOB RDAREA be allocated to a single LOB global buffer.

Operand rules

- If the name of an RDAREA includes characters other than single-byte alphanumeric characters, the name must be enclosed in quotation marks (").
- When an RDAREA name is enclosed in quotation marks ("), the name becomes case sensitive (lowercase characters are distinguished from uppercase characters). However, if an RDAREA name is not enclosed in quotation marks, all the characters are handled as uppercase characters.
- A maximum of 4,096 RDAREAs can be defined for a single global buffer.

Notes

LOB global buffers do not use the prefetch facility or the deferred write trigger facility. Therefore, the `-m, -p,` and `-w` options need not be specified.

-o

Specifies that this global buffer is to be allocated to all RDAREAs that are not specified in the `-r` option. The `-o` option can be specified only once. If it is specified more than once, the first specification is used.

-i *authorization-identifier.index-identifier*

***authorization-identifier* ~<identifier>((1-8 characters))**

***index-identifier* ~<identifier>((1-30 characters))**

Specifies the name (*authorization-identifier.index-identifier*) of an index to which the global buffer is to be allocated as an index global buffer.

Specification guidelines

Specify an index that is used frequently. When a global buffer is allocated to a frequently used index, the memory residency of the index page increases, and as a result, the number of inputs and outputs can be reduced.

This effect can be especially large if an index defined as a cluster key or unique key is allocated to the global buffer. Note that because the index identifier of a cluster key is determined by HiRDB, check the index identifier by searching the dictionary table (`INDEX_NAME` column of the `SQL_INDEXES` table) after a table has been defined.

If all indexes are allocated to the global buffer for indexes, the overall efficiency of the global buffer declines. Therefore, carefully select the indexes to be allocated to the global buffer according to the memory size.

Operand rules

- Each *authorization-identifier.index-identifier* to be specified must be unique.
- When an authorization identifier or index identifier includes characters other than single-byte alphanumeric characters, the authorization identifier or index identifier must be enclosed in quotation marks (").
- When an authorization identifier or index identifier is enclosed in quotation marks ("), the name becomes case sensitive (lowercase characters are distinguished from uppercase characters). However, if an authorization identifier or index identifier is not enclosed in quotation marks, all the characters are handled as uppercase characters.

Notes

- A global buffer must also be allocated to the RDAREA that stores the index specified in the `-i` option. Either specify an RDAREA in the `-r` option or allocate a global buffer by using the `-o` option.

  RDAREAs store the directory section (directory pages), the data section (table-data pages), and the index section (index pages). Because only the index section is cached in the index global buffer, a global buffer with `-r` or `-o` specified is needed to cache the directory and data sections.
- If `DROP TABLE` or `DROP INDEX` is executed, the index global buffer allocated to the deleted index will no longer be used. If an index having the same name as the deleted index is defined, its index global buffer cannot be used. To use that index global buffer, normally terminate HiRDB, and then restart it.
- If a nonexistent index is specified in the `-i` option when HiRDB starts, its index global buffer is not allocated. In this case, the `KFPH23014-W` message is issued.

  If the index specified in the `-i` option is defined after HiRDB has started, its index global buffer cannot be used. To use that index global buffer, normally terminate HiRDB, and then restart it.
- An index global buffer allocated to a temporary table index is disabled when HiRDB starts. In this case, the `KFPH23014-W` message is issued.
- If an index storage RDAREA whose page size is larger than the size of the index global buffer is added with `ALTER TABLE`, the index pages in the added RDAREA are not cached in the index global buffer. To cash those index pages in the index global buffer, normally terminate HiRDB, and then restart it.
- If the page size of the RDAREA is made larger than the size of the index global buffer by using the RDAREA re-initialization feature of the database structure modification utility, the index pages of the re-initialized RDAREA are not cached in the index global buffer. To cache those index pages in the index global buffer, normally terminate HiRDB, and then restart it.

-n *buffer-sectors-count*

~<unsigned integer>

- 32-bit mode: **((4-460000))**
- 64-bit mode: **((4-1073741824))**

Specifies a sectors count for the global buffer. This option is required.

Specification guidelines

- Output operations to the database might become concentrated depending on the synchronization point acquisition timing and the updated pages rate in the global buffer. Therefore, the balance of I/O operations must be taken into consideration as well.
- The following table can be used to determine an appropriate global buffer sectors count:

Condition		Appropriate global buffer sectors count
Global buffer to which specification of `-r` or `-o` option is applicable[#1]	HiRDB/Single Server	*Number of concurrently occurring SQL processing requests × number of pages used by one SQL statement (roughly 3 to 6)*
	HiRDB/Parallel Server	N*umber of concurrently executing users × average number of concurrently accessed tables per transaction ×* 3 × *n*[#2]

Condition	Appropriate global buffer sectors count
Global buffer to which specification of `-b` option is applicable	*Number of data pages stored in LOB RDAREA* × *residency degree*[#3] (%)
Global buffer to which specification of `-i` option is applicable[#1]	*Number of pages storing index* × *residency degree*[#4] (%)

#1

The minimum number of buffer sectors necessary is explained below. Specify a value that at least equals this value for the buffer sector count. If a value smaller than this value is specified, a buffer shortage might cause an SQL error.

- HiRDB/Single Server
  number-of-SQL-process-requests-that-occur-concurrently × 4
- HiRDB/Parallel Server
  number-of-concurrently-executing-users × *average-number-of-concurrently-accessed-tables-inside-each-transaction* × 4

#2

A buffer sectors count derived with $n = 1$ cannot increase the buffer hit rate. A count needs to be specified that takes into consideration how many extra sectors the coefficient needs to provide (how high of a buffer hit rate you want).

#3

Although a residency degree of 1 (100%) is desirable, the value that is used must take into account the memory capacity, data access frequency, and so on. For details about the total number of pages in a user LOB RDAREA and the total number of pages in a registry RDAREA, see the *HiRDB Version 9 Installation and Design Guide.*

#4

Although a residency degree of 1 (100%) is desirable, the value that is used must take into account the memory capacity and importance of the index. For details about the number of pages for storing an index, see the *HiRDB Version 9 Installation and Design Guide.*

Tuning the specified value

Set a value such that the buffer hit rate is at least 80% for HiRDB jobs. The buffer hit rate can be determined as follows:

- With the statistics analysis utility (check the *updated buffer hit rate and referenced buffer hit rate* in the statistical information on the global buffer)
- By checking the header `HIT` from the `pdbufls` command.

Notes

- If an unnecessarily large value is specified, the number of inputs/outputs will decrease but the overhead for buffer retrieval will increase.
- Because global buffers are allocated in the shared memory, allocating an unnecessarily large area will cause frequent paging during use of other memory, resulting in degraded performance.
- Note that if the number of buffer sectors is too large, it might not be possible to allocate shared memory.
- If the number of users accessing the same global buffer increases and a shortage occurs in the number of buffer sectors, an SQL error might occur.

-l *buffer-size*

~<unsigned integer>((even number between 4 and 30))(kilobytes)

Specifies, in kilobytes, the size of one global buffer sector. Specify this option with the `-n` option.

Specification guidelines

Normally, omit this option. If this option is omitted, the maximum page size of the RDAREA allocated to this global buffer is used as the buffer size. For a HiRDB/Parallel Server, the maximum page size of the RDAREA inside each server is used as the buffer size, and consequently, the buffer size might be different for each server.

In the following cases, consider changing the specification value:

- Set a value on the larger side when an RDAREA with a page size exceeding the value that would be set in this option will be added later or the maximum RDAREA page size will be increased through reinitialization. However, if HiRDB can be stopped, there is no need to change the specification value because the maximum page size will be set for the buffer size during the subsequent HiRDB startup.

Operand rules

- If the value specified for this option is smaller than the maximum page size of the RDAREAs, the latter value will be used as the buffer size.
- If an odd number is specified for this option, the actual buffer size will be 1 greater than the specified value.

Note

When the value of this option is changed, the new value will not go into effect until HiRDB is started normally. When HiRDB is restarted, the buffer size that was in effect during the last operation (the size before the change) is used.

-m *maximum-concurrently-executable-prefetches-count*

~<unsigned integer>((0-95000))<<0>>

Specifies the maximum number of prefetch facilities that can be used concurrently. The prefetch facility can reduce the input/output time when a large volume of data is retrieved using the raw I/O facility.

When `0` is specified or this option is omitted, the prefetch facility will not work. To use the prefetch facility, a value of at least 1 must be specified.

Specification guidelines

Specify the number of times an SQL statement or utility, to which the prefetch facility will be applied, can be executed concurrently in the RDAREA table allocated to this global buffer. The prefetch facility loads pages in the batch mode for the following SQL statements or utility:

1. Data pages are loaded in the batch mode for the `SELECT`, `UPDATE`, or `DELETE` statement that does not use an index.
2. Index leaf pages are loaded in the batch mode for the `SELECT`, `UPDATE`, or `DELETE` statement (excluding = condition and `IN` condition) that performs ascending-order retrieval# using an index.
3. Index leaf pages and data pages are loaded in the batch mode for the `SELECT`, `UPDATE`, or `DELETE` statement (excluding = condition and `IN` condition) that performs ascending-order retrieval# using a cluster key.
4. Index leaf pages and data pages are loaded in the batch mode for the unload processing of the database reorganization utility that does not use a local buffer.

#: In the order specified in the index definition in the case of a multicolumn index

Notes

When the prefetch facility is used, a buffer dedicated for batch input is allocated in the shared memory for global buffers, separately from the global buffers. Consequently, the size of the shared memory must be estimated again.

If the SQL statements or utilities to which the prefetch facility is applied are concurrently executed in a table in the RDAREA allocated to the global buffer and any of the SQL statements or utilities exceeds the specified execution count, the prefetch facility is no longer applied to that SQL statement or utility.

For details about the prefetch facility and the formula for determining the size of the shared memory to be used by global buffers, see the *HiRDB Version 9 Installation and Design Guide.*

-p *maximum-batch-input-pages-count*

~<unsigned integer>((2-256))<<32>>

Specifies the maximum number of pages that can be input in a batch by the prefetch facility. This option is valid only when a value of at least 1 is specified for the `-m` option.

Specification guidelines

The value to be specified is based on the size of the shared memory and its reduction cost performance so that it satisfies the following formula:

$a \times b$ = 64-128 (kilobytes)

a: Page size of the RDAREA that stores the data or index to be prefetched.

b: Maximum number of pages to be input in a batch.

-w *updated-pages-output-rate-during-deferred-write-trigger*

~<unsigned integer>((0-100))<<20>>(%)

Specifies as a percentage the updated pages output rate at a deferred write trigger. For details about the deferred write trigger, see the *HiRDB Version 9 Installation and Design Guide.*

If `0` is specified in this option, updated pages are not output at a deferred write trigger.

Specification guidelines

- The value to be specified is determined by using the statistics analysis utility to check the number of inputs/outputs and the updated pages hit rate for each global buffer. Specify a low output rate for a global buffer with a high updated pages hit rate, and specify a high output rate for a global buffer with a low updated pages hit rate.
- If an unnecessarily large value is specified, frequent updates will result in a large number of input/output operations. Moreover, the number of pages that must be written into the database during delayed write will increase, reducing the throughput. Conversely, if an unnecessarily small value is specified, the number of pages that must be written into the database during a synchronization point dump might increase. Select a value that minimizes the number of times the same page is written into the database during synchronization point dump output intervals.
- If all pages are in the global buffer and multiple transactions update the same page frequently, specify `0` for this operand. Specifying `0` can reduce the total number of pages that are output to the database within the synchronization point interval.

-c

This option is related to the standby-less system switchover facility. If you will not be using this facility, do not specify this option.

- Standby-less system switchover (1:1) facility

  Specify this option if you create a global buffer that will be used by the RDAREA or indexes for the normal BES and also on the target alternate BES. If this option is omitted, no global buffer is created on the alternate BES. In this case, the global buffer with the `-o` option specified that is defined for the alternate BES is used during alternation.

  Note that if you omit both the `-c` and `-o` options, an alternate BES unit cannot be started.

  The specification for this option determines how alternating global buffers are allocated. For details about how to allocate global buffer during alternation, see *Definition of global buffers (standby-less system switchover (1:1) facility only)* in the *HiRDB Version 9 System Operation Guide*.
- Standby-less system switchover (effects distributed) facility

  Specify this option to define global buffers for the RDAREAs or indexes located on the back-end server that is subject to standby-less system switchover. The global buffers defined here are created on all units in the same HA group.

  If the `-c` option is omitted, an RDAREA that uses this global buffer cannot be accessed because this global buffer is not allocated.

  The specification of this operand determines the method of allocating global buffers to the RDAREAs or indexes of the back-end server that is the target of the standby-less system switchover (effects distributed) facility. For details about how to allocate global buffers when the standby-less system switchover (effects distributed) facility is being used, see the *HiRDB Version 9 System Operation Guide*.

-y *update-buffer-sectors-count-for-deferred-write-trigger-event*

~<unsigned integer>((2-2147483647))

Specifies the deferred write trigger in terms of a number of update buffer sectors. When the number of update buffer sectors reaches the specified value, the updated pages are written to the disk. Use this option to set the deferred write trigger for each global buffer.

Specification guidelines

Normally omit this operand. Deferred write processing might not always be completed within the synchronization point dump acquisition interval. You can specify this operand for such cases if you want to shorten the writing time by reducing the number of updated buffers and reducing slightly the updated buffer hit rate. A guideline for this operand's value is to use 50% (the initial value set by HiRDB) or determine the operand's value by referring to *Tuning the synchronization point processing time when deferred write processing is used* in the *HiRDB Version 9 System Operation Guide*.

Operand rules

If the value specified in this option is greater than the number of global buffer sectors, the number of the global buffer sectors is used.

Notes

If the value of this option is too small, the number of times deferred write processing is executed increases, resulting in an increase in the workload; therefore, it is recommended that you specify an appropriate value in accordance with the specification guidelines.

Relationships to other operands

This operand has the following relationships with the `pd_dbbuff_rate_updpage` operand:

- The value set for the `pd_dbbuff_rate_updpage` operand applies to all the global buffers.
- The value of the `pdbuffer` operand's `-y` option applies to each global buffer.
- The `pdbuffer` operand's `-y` option takes precedence.
- If the `pdbuffer` operand's `-y` option is omitted, the number of update buffer sectors for deferred write trigger event depends on the specification of the `pd_dbbuff_rate_updpage` operand, as shown below:

pd_dbbuff_rate_updpage operand specification	Number of update buffer sectors for deferred write trigger event
Specified	Number of global buffer sectors × `pd_dbbuff_rate_updpage` operand value
Omitted	Determined automatically by HiRDB

Effects on individual estimation formulas

If the value of the `pdbuffer` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*
- *Formula for size of shared memory used by global buffers*
- *Formula 1*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for shared memory used by a single server*
- *Formula 2*, *Formula 3*, *Formula 4*, and *Formula 5* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 2*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for the size of the shared memory used by a back-end server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

2.2.47 Operands related to HA groups

239) pdhagroup -g *HA-group-identifier* -u unit-identifier[,*unit-identifier*]...

When you are using the standby-less system switchover (effects distributed) facility, use this operand to define an *HA group*.

There is no upper limit on the number of HA groups that can be defined on a single system.

For details about HA groups, see the *HiRDB Version 9 System Operation Guide*.

-g *HA-group-identifier*

~<identifier>((1-8 characters))

Specifies an HA group identifier. Specify an identifier that is unique to an HA group within the system.

-u *unit-identifier*[,*unit-identifier*]...

~<identifier>((4 characters))

Specifies the unit identifier for the unit that comprises an HA group. The number of units that can be specified is between 2 and 32. Specifying duplicate units causes an error. The units to be specified in this option must satisfy all of the following conditions:

- All units within the HA group belong to the same network segment.
- The unit does not belong to another HA group.
- The standby-less system switchover (effects distributed) facility is being applied. (`pd_ha_agent = activeunits` is specified.)
- At least one host back-end server exists inside the unit. (An accepting-only unit cannot be defined.)
- The number of host back-end servers within the unit plus the maximum number of guest back-end servers that can be accepted (value specified for the `pd_max_act_guest_servers` operand) does not exceed 34.
- The unit is comprised of back-end servers only.
- All servers within the unit belong to an HA group.

Effects on individual estimation formulas

If the value of the `pdhagroup` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

2.2.48 Operands related to statistical information

240) pdstbegin

Operand specification format in a HiRDB/Single Server

`pdstbegin [-k` *statistical-information-type*`[,`*statistical-information-type*`]...]`

`[-m` *interval*`]`

`[-w]`

Operand specification format in a HiRDB/Parallel Server

`pdstbegin [-k` *statistical-information-type*`[,`*statistical-information-type*`]...]`

`[-m` *interval*`]`

`[{-x` *host-name* `| -u` *unit-identifier*`}]`

`[{-a | -s` *server-name*`[,`*server-name*`]...}]`

`[-w]`

This operand is specified in order to begin collecting statistical information at the time HiRDB is started. The `pdstend` command is entered in order to stop collection of statistical information. The statistical information is output to the statistics log file.

-k *statistical-information-type*[,*statistical-information-type*]...

~<<sys>>

Specifies the type of statistical information that is to be output. The `pdls -d sty` command can be used to check the types of statistical information specified for output.

Statistical information type (-k option specification)	Type of statistical information output	Statistical information output trigger
`sys`	Statistical information on system operation	Statistical information is output at the interval specified in the `-m` option.
`uap`	Statistical information on UAPs	Statistical information is output during connection to and disconnection from HiRDB.

Statistical information type (-k option specification)	Type of statistical information output	Statistical information output trigger
`sql`	Statistical information on SQLs	Statistical information is output when SQL execution starts and ends.
`buf`	Statistical information on global buffers[#1]	Statistical information is output at synchronization points.
`fil`	Statistical information on HiRDB files related to database manipulations	
`dfw`	Statistical information on deferred write processing	
`idx`	Statistical information on indexes	
`sop`	Statistical information on SQL static optimization	Statistical information is output when no hit occurs on an SQL object buffer during preprocessing of a dynamic SQL or static SQL.
`dop`	Statistical information on SQL dynamic optimization	Statistical information is output when an SQL statement other than `FETCH` or `CLOSE` is executed.
`pcd`	Statistical information related to SQL object execution	Statistical information is output when an SQL object is executed.
`sqh`	Statistical information on SQL statement statistics[#2]	Statistical information is output during `PREPARE` and when an embedded SQL is executed.
`obj`	Statistical information on SQL object transmission[#3]	Statistical information is output when an SQL object is executed.

#1: This statistical information is collected at synchronization points, and the information between synchronization points is edited. Consequently, statistical information cannot be collected unless at least two synchronization points occur. To reliably collect this statistical information, a synchronization point must be triggered by executing the `pdlogswap` or `pdlogsync` command immediately before executing the `pdstend` command.

#2: The statistical information on SQL statement statistics is output when the statistical information on SQLs (`sql` specification) is being output.

#3: This statistical information is only for a HiRDB/Parallel Server, and is not output for a HiRDB/Single Server.

-m interval

~<unsigned integer> ((1-1440)) <<10>>(minutes)

This option is specified when statistical information on system operation (sys specified) is to be collected.

Specifies in minutes the interval at which the statistical information on system operation is to be output to the statistics log file.

-x *host-name*

~<host name> ((1-32 characters))

For a HiRDB/Parallel Server, specifies a host name specified in the `-x` option of the `pdunit` operand in order to select the units for which statistical information is to be output. Specify the host name of the unit for which statistical information is to be output.

If the standby-less system switchover (effects distributed) facility is being applied to this unit, the statistical information on the guest back-end server being accepted is also output.

-u *unit-identifier*

~<identifier> ((4 characters))

In the case of a HiRDB/Parallel Server, specifies the unit name of a unit for which statistical information is to be output. This option is specified when it is necessary to limit the units for which statistical information is to be output.

If the standby-less system switchover (effects distributed) facility is being applied to the unit on this host, the statistical information on the guest back-end server being accepted is also output.

-a

In the case of a HiRDB/Parallel Server, specifies that statistical information is to be output for all servers. This option is specified when it is not necessary to limit the units for which statistical information is to be output.

-s *server-name*[,*server-name*]...

~<identifier> ((1-8 characters))

In the case of a HiRDB/Parallel Server, specifies the server names of the servers for which statistical information is to be output. This option is specified when it is necessary to limit the servers for which statistical information is to be output. Some types of statistical information are not output for all server types, as indicated in the following table:

Statistical information type (-k option specification)	Server type		
	FES	DS	BES
`sys`	Y	Y	Y
`uap`	Y	--	--
`sql`	Y	--	--
`buf`	--	Y	Y
`fil`	--	Y	Y
`dfw`	--	Y	Y
`idx`	--	Y	Y
`sop`	Y	--	--
`dop`	Y	--	--
`pcd`	Y	Y	Y
`sqh`	Y	--	--
`obj`	--	Y	Y

Y: Statistical information is output.

--: Statistical information is not output.

-w

Specifies that statistical information about the thread-to-thread lock-release wait time is to be output. This option is enabled when statistical information about system activities is output. Therefore, statistical information about the thread-to-thread lock-release wait time can be acquired if the `-k` option is omitted or the specified `-k` option contains `sys`.

This option might affect the performance of the entire system; therefore, we recommend that you normally do not specify this option.

Specification guideline

If you use the system switchover facility, we recommend that you specify this operand to collect thorough statistical information.

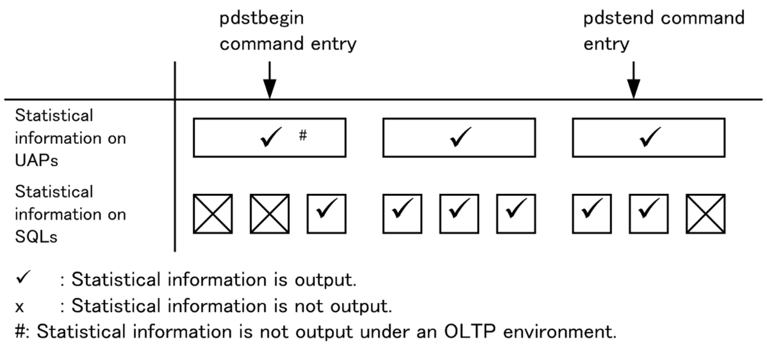
Relationship to other operands

- This operand is related to `pd_stj_file_size`.
- When this operand is specified, specification of the `pd_statistics` operand is ignored.

Notes

- The `pdstbegin` operand can be specified only once. If it is specified more than once, the first time it is specified is valid and the subsequent specifications are ignored.
- When the `pdstbegin` operand is specified, the specified statistical information is output until HiRDB is terminated or until the `pdstend` command is entered.
- If you use the standby-less system switchover (1:1) facility, the specification for this operand is invalid when the system is being switched to an alternate BES unit.

- Note the following if you use the standby-less system switchover (effects distributed) facility: During a normal HiRDB startup, each server follows the value specified for this operand. During a HiRDB restart (including a system switchover), the value specified for this operand is invalid. During a restart, the previous statistical information collection state is inherited. During a system switchover, the statistical information prior to the switchover is inherited.
- If no server has started in the unit, unit statistical information is not acquired.
- Depending on when the `pdstbegin` and `pdstend` commands are entered, the statistical information on UAPs might not match the statistical information on SQLs. The relationship between when the `pdstbegin` and `pdstend` commands are entered and the statistical information that is output is shown as follows:


pdstbegin command entry
pdstend command entry
Statistical information on UAPs
Statistical information on SQLs
✓ : Statistical information is output.
x : Statistical information is not output.
#: Statistical information is not output under an OLTP environment.

- If the HiRDB system, unit, or server is terminated (including abnormal termination) and is then started again, the statistical information collection state might not be inherited. The following table shows whether the statistical information collection state is inherited when the HiRDB system, unit, or server is started.

Start mode	Statistical information collection environment	Start condition			
		HiRDB start	Unit start	Server start	
				Standby-less system switchover (effects distributed) facility is not used	Standby-less system switchover (effects distributed) facility is used
Normal start	Statistical information is being collected with the `pdstbegin` operand specified.	Y	Y	N#	Y
	Statistical information is being collected by executing the `pdstbegin` command.	N#	N#	N#	N#
Restart	Statistical information is being collected with the `pdstbegin` operand specified.	Y	Y	--	I
	Statistical information is being collected by executing the `pdstbegin` command.	N#	N#	--	Y

Y: Statistical information collection state is inherited.

I: Statistical information collection state is inherited. When the `pdstbegin` command is executed, the collection state of the statistical information specified by this command is inherited.

N: Statistical information collection state is not inherited.

--: Not applicable.

#: To collect statistical information, you must execute the `pdstbegin` command after starting the HiRDB system.

241) pdhibegin -k *statistics-type*[,*statistics-type*]...

Specifies the types of statistical information to be collected beginning at the time of HiRDB startup. When this operand is specified, statistical information is output continuously to the system log file until HiRDB is terminated.

-k *statistics-type*

Specifies the type of statistical information to be output:

`cnc`: Statistical information related to CONNECT/DISCONNECT.

In the case of a HiRDB/Parallel Server, the output destination file is the system log file at a front-end server.

2.2.49 Operands related to a client group

242) pdcltgrp -g *client-group-name -u guaranteed-number-of-connected-users-per-group*

This operand is specified when the connection frame guarantee facility for a client group is used. For details about the connection frame guarantee facility for a group, see the *HiRDB Version 9 System Operation Guide.*

-g *client-group-name* ~<alphabetics>((1-2 characters))

Specifies the name of a client group.

Client type	Client group name	Remarks
X/Open XA interface	XA#1	Client that accesses HiRDB via the X/Open XA interface. Even if the client is a PC or WS, the client group is XA as long as the X/Open XA interface is used for accessing HiRDB.
PC client	PC#2	Windows and Linux clients
WS client	WS#2	UNIX client
Mainframe client	MF	VOS3 client
User-defined client group#3	One uppercase character#1, #4	--

Legend:

--: Not applicable.

#1: To specify this client group, the HiRDB client version must be one of the following:

- 04-05 (cannot be specified for 05-00 or 05-01)
- 05-02 or later

#2: To specify this client group, the HiRDB client version must be the following:

- 04-00 or later

#3: Up to 10 user-defined groups can be specified.

#4: The client group name specified here is specified for the `PDCLTGRP` operand of the client environment definition. If the values specified for the `pdcltgrp` and `PDCLTGRP` operands do not match, the value specified here is ignored.

-u *guaranteed-number-of-connected-users-per-group*

~<unsigned integer>

- For a HiRDB/Single Server: ((1-2999))
- For a HiRDB/Parallel Server: ((1-1999))

Specifies the number of connected users for the client group.

Even when accesses to HiRDB from other client groups or utilities are concentrated, the users (up to the number specified here) from the applicable client group are guaranteed access to HiRDB.

Notes

- The total number of users specified by the `-u` option must not exceed the value specified in the `pd_max_users` operand; otherwise, HiRDB will not start.
- This facility is applicable only to the database definition utility.

Effects on individual estimation formulas

If the value of the `pdcltgrp` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

2.2.50 Operands related to plug-ins

243) pdplugin -n *plug-in-name*

~<identifier>((1-30 characters))

Specifies the name of a plug-in to be used in HiRDB. This operand is omitted when no plug-ins are to be used.

For the names of the plug-ins that can be specified here, see the manuals for the respective plug-ins.

Conditions

A plug-in specified here must have been registered into HiRDB in advance with the `pdplgrgst` command.

2.2.51 Operands related to shared memory

244) SHMMAX *maximum-shared-memory-segment-siz*e

~<unsigned integer> (megabytes)

- 32-bit mode: **((6-2047)) <<200>>**
- 64-bit mode: **((6-4194304)) <<1024>>**

Specifies, in megabytes, the maximum segment size for the shared memory for global buffer.

HiRDB allocates shared memory segments for global buffers up to the size specified for this operand. If the total size of the global buffers allocated to RDAREAs on the server machine exceeds the value specified for this operand, multiple shared memory segments are allocated.

When HiRDB starts, HiRDB can allocate a maximum of 16 shared memory segments for global buffers per server. Note that, in the 32-bit mode, the maximum number of shared memory segments that can be allocated when HiRDB starts is the value specified for the `pd_max_dbbuff_shm_no` operand. You can use the `pdls -d mem` command to obtain information about the shared memory segments used for global buffers. For details about the `pdls -d mem` command, see the manual *HiRDB Version 9 Command Reference*.

Specification guidelines

- For 32-bit mode

  Normally, this operand does not need to be specified. Specify this operand only if HiRDB startup fails with the `KFPH23005-E` message (error code `24`) issued. For details, see the handling of the `KFPH23005-E` message.

  If the `KFPH23005-E` message displays error code `20`, take the appropriate action by referencing the manual *HiRDB Version 9 Messages*.

  If the facility for dynamically changing global buffer is used and the value of the `SHMMAX` operand is too large, the restart processing might fail because the required area cannot be allocated. To avoid this, specify a value of 50 or smaller in the `SHMMAX` operand.

- For 64-bit mode

  If the size of shared memory for global buffers is 16 gigabytes or less, there is no need to specify this operand.

  If the size of shared memory for global buffers is greater than 16 gigabytes, specify the value that satisfies the following condition:

  SHMMAX operand value × 16 ≥ *size of shared memory for global buffers*

Operand rule

If the specified value extends beyond the permitted range, the default value is assumed.

Notes

- For SHMMAX, specify a value that is larger than the size of the area for managing the shared memory used for the global buffers. If a larger value is not specified, HiRDB cannot be started. For details about the formula for determining the size of the area for managing the shared memory used for the global buffers, see *Formula for size of shared memory used by global buffers* in the *HiRDB Version 9 Installation and Design Guide*.
- If the facility for dynamically changing global buffer is used, the value of the SHMMAX operand can be changed only after normal termination. If the KFPH23005-E message with error code 20 is issued after forced termination or during restart following planned termination, change the specified value by referencing item 3 in the table provided for error code 20 in the KFPH23005-E message in the manual *HiRDB Version 9 Messages*.
- If the shared memory is to be allocated to a file under the HiRDB directory, but there is not enough free space on the disk containing the HiRDB directory, an insufficient disk capacity error occurs.
- If the shared memory is to be allocated to a page file, but the page file capacity is not sufficient, an insufficient page file capacity error occurs.
- If you use the facility for fixing shared memory pages, specify a value in this operand that is an integral multiple of the page size of a Windows large page. Shared memory pages are fixed in the following cases:
  - The value fixed is specified in the pd_dbbuff_attribute operand.
  - A value of 1 or greater is specified in the pd_max_resident_rdarea_no operand.

  If the specified value is not an integral multiple of the page size, HiRDB increases it to the next integral multiple of the page size.
- If the value of the SHMMAX operand is changed, re-estimate the pd_max_dbbuff_shm_no operand value and the number of resources (amount of shared memory) required for the PDUXPLSHMMAX system environment variable.

Relationship to other operands

This operand is related to the following operands:

- pd_dbbuff_modify
- pdbuffer
- pd_max_add_dbbuff_no
- pd_max_add_dbbuff_shm_no
- pd_sysdef_default_option (If v6compatible is specified for HiRDB for 32-bit mode, the default value of the SHMMAX operand is 6.)
- pd_max_dbbuff_shm_no
- PDUXPLSHMMAX system environment variable

Effects on individual estimation formulas

If the value of the SHMMAX operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Memory required by in-memory data processing* under *Estimating the memory size required for a HiRDB/ Single Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Single Server*
- *Memory required by in-memory data processing* under *Estimating the memory size required for a HiRDB/ Parallel Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

2.2.52 Operands related to date and time

245) TZ *time-zone*

~<character string>

Specifies the time zone to be applied to the date and time of the log to be output. Time zone means the environment variable in which the environment for displaying the time is specified.

Make sure that the value of this operand (default: `JST-9`) is the same as the value of the `TZ` system environment variable for the server machine. To display the server machine's environment variables, enter the `set` command at the command prompt. If the `TZ` system environment variable is undefined or its value is different from this operand value, HiRDB operation cannot be guaranteed.

2.2.53 Operands related to the message output suppression facility

246) pdmlgput -s *output-selection*

{-c ALL | [-l *message-severity*]

-m *message-ID*[,*message-ID*]...}

Specifies options for HiRDB's output of messages to the the event log. The following can be specified:

- Suppress output of all messages
- Output specified messages only
- Change the severity level of the messages that are output

When this operand is not specified, output of messages to the event log is not suppressed.

-s *output-selection*

Specifies the message output suppression option for the indicated messages.

`Y`: Outputs the indicated messages.

`N`: Suppresses output of the indicated messages.

For details about suppression of message output to the the event log, see the *HiRDB Version 9 System Operation Guide*.

-c ALL

Specifies that output of all HiRDB message to the the event log is to be suppressed.

When this option is specified, specify `N` for the `-s` option.

Note

Normally, it is recommended that this option not be specified, because it makes it difficult to diagnose errors.

-l *message-severity*

Specifies the severity level to be set for messages that are output.

To specify this option, you must also specify `Y` in the `-s` option.

`E`: Error message

`W`: Warning message

`Q`: Waiting message

`I`: Information message

For details about changing the severity level of messages that are output to the the event log, see the *HiRDB Version 9 System Operation Guide*.

-m *message-ID*[,*message-ID*]...

Specifies the IDs of the messages that are to be output or that are to be suppressed. Do not enter the message severity level that follows the hyphen (-) in a message ID. (For example, for `KFPS01820-E`, specify `KFPS01820`.)

Specifying a message that is not designed to be output in the event log for this option is the same as not specifying it.

Operand rules

- You can specify multiple lines of this operand.

- If multiple lines of control are specified for the same message using the `ALL` specification or message ID specification, the latter specification takes precedence.

Relationship to other operands

- `pd_mlg_file_size`: When message output is suppressed, specifies the maximum size of the message log file.
- `pd_mlg_msg_log_unit`: When messages are to be output, specifies the output destination for message logs (the system manager unit or the unit that outputs each message).

3 Unit Control Information Definition

This chapter explains the operands of the unit control information definition.

3.1 Operand formats

A unit control information definition defines information about a unit. This section explains the formats used to specify the operands of a unit control information definition. Note that the numbers in the following table correspond to the numbers assigned to the explanations of the individual operands in *3.2 Operand explanations*.

For users who are creating HiRDB system definitions for the first time

The first step is to determine the values to be specified for the operands shown in boldface type. In principle, HiRDB can be started once the boldface operands have been specified.

For users of the standby-less system switchover (effects distributed) facility

When you use the standby-less system switchover (effects distributed) facility, you can specify only certain operands. For details, see *F. List of Operands That Can Be Specified When Using the Standby-less System Switchover (Effects Distributed) Facility (Unit Control Information Definition)*.

No.	Format	Operand category
1	**set pd_unit_id =** ***unit-identifier***	System structure
2	[set pd_hostname = *host-name*]	
3	[set pd_max_server_process = *maximum-number-of-concurrently-activated-server-processes*]#	Maximum concurrent executions
4	[set pd_server_entry_queue = spnfifo | fifo | loop]#	HiRDB processing
5	[set pd_thdlock_wakeup_lock = Y | N]#	
6	[set pd_thdlock_pipe_retry_interval = *thread-lock-release-check-interval*]#	
7	[set pd_thdlock_retry_time = *thread-lock-sleep-time*]#	
8	[set pd_thdspnlk_spn_count = *thread-spin-lock-spin-count*]#	
9	[set pd_db_io_error_action = dbhold | unitdown]#	
10	[set pd_max_recover_process = *concurrently-executable-full-recovery-processes-count*]#	Full recovery processing
11	[set pd_trn_rcvmsg_store_buflen = *transaction-recovery-message-queue-size*]#	Transaction decision processing
12	[set pd_watch_time = *maximum-response-wait-time*]#	System monitoring
13	[set pd_down_watch_proc = *upper-limit-for-server-process-abnormal-terminations*[,*monitoring-interval*]]#	
14	[set pd_cwaittime_report_dir = *SQL-runtime-warning-information-file-output-destination-directory*]*	SQL runtime warning output facility
15	[set pd_cwaittime_report_size = *SQL-runtime-warning-information-file-maximum-size*]*	
16	[set pd_uap_exerror_log_dir = *SQL-error-report-file-storage-directory*]#	Facility for output of extended SQL error information
17	[set pd_uap_exerror_log_size = *SQL-error-report-file-maximum-size*]#	
18	[set pd_uap_exerror_log_param_size = *maximum-data-size-of-parameter-information-to-be-output-to--client error-log-file-and-SQL-error-report-file*]#	
19	[set pd_lck_wait_timeout = *lock-release-wait-time*]#	Lock
20	[set pd_lck_release_detect = interval | pipe]#	
21	[set pd_lck_release_detect_interval = *lock-release-detection-interval*]#	

No.	Format	Operand category
22	`[set pd_lck_deadlock_info = Y | N]`#	
23	`[set pd_lck_deadlock_check = Y | N]`#	
24	`[set pd_lck_deadlock_check_interval =` *deadlock-check-interval*`]`#	
25	`[set pd_thread_stack_expand_size =` *stack-extension-size-per-thread*`]`	Buffers
26	`[set pd_shmpool_attribute = free | fixed]`#	Shared memory
27	`[set pd_dbbuff_attribute = free | fixed]`#	
28	`[set pd_stj_file_size =` *maximum-statistics-log-file-size*`]`#	Statistical information
29	`[set pd_stj_buff_size =` *statistics-log-buffer-size*`]`#	
30	`[set pd_rpc_trace = Y | N]`#	RPC trace information
31	`[set pd_rpc_trace_name = "`*name-of-RPC-trace-collection-file*`"]`#	
32	`[set pd_rpc_trace_size =` *RPC-trace-collection-file-size*`]`#	
33	`[set pd_prf_level = 00000007|0000001f|0000007f|000001ff|00000000]`#	Performance trace information
34	`[set pd_prf_file_count =` *number-of-performance-trace-information-file-generations*`]`#	
35	`[set pd_prf_file_size =` *size-of-a-performance-trace-information-file*`]`#	
36	`[set pd_cancel_dump = put | noput]`#	Troubleshooting information
37	`[set pd_dump_suppress_watch_time =`*troubleshooting-information-output-suppression-time*`]`#	
38	`[set pd_spool_cleanup_interval =` *troubleshooting-information-deletion-interval*`]`#	
39	`[set pd_spool_cleanup_interval_level =` *number-of-days*`[,`*deletion-type*`]]`#	
40	`[set pd_spool_cleanup = normal | force | no]`#	
41	`[set pd_spool_cleanup_level =` *number-of-days* [,*deletion-type*`]]`#	
42	`[set pd_module_trace_max =` *maximum-number-of-module-traces-that-can-be-stored*`]`#	
43	`[set pd_module_trace_timer_level = 0 | 10 | 20]`#	
44	`[set pd_pth_trace_max =` *maximum-number-of-stored-communication-traces*`]`#	
45	`[set pd_dbbuff_wait_interval =` *global-buffer-occupation-state-check-interval*`]`#	Global buffers
46	`[set pd_dbbuff_wait_spn_count =` *maximum-spin-loop-count-for-global-buffer-occupation-state-checking*`]`#	
47	**set pd_syssts_file_name_1 = "*logical-file-name*" ,"*file-a-status-file-name*" ,"*file-b-status-file-name*"**	Unit status files
	[set pd_syssts_file_name_2 = "*logical-file-name*" ,"*file-a-status-file-name*" ,"*file-b-status-file-name*"]	
	`[set pd_syssts_file_name_3 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_syssts_file_name_4 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	

No.	Format	Operand category
	`[set pd_syssts_file_name_5 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_syssts_file_name_6 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_syssts_file_name_7 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-fil*e-name`"]`	
48	`[set pd_syssts_initial_error = stop | continue | excontinue]`	Unit status files (when an error occurs)
49	`[set pd_syssts_singleoperation = stop | continue]`	
50	`[set pd_syssts_last_active_file = "`*logical-file-name*`"]`	
51	`[set pd_syssts_last_active_side = A | B]`	
52	`[set pd_audit = Y | N]`#	Security
53	`[set pd_aud_file_name = `*HiRDB-file-system-area-name-for-audit-trail-file*`]`#	
54	`[set pd_aud_max_generation_size = `*audit-trail-file-maximum-size*`]`#	
55	`[set pd_aud_max_generation_num = `*maximum-audit-trail-file-count*`]`#	
56	`[set pd_aud_async_buff_size = `*size-of-buffer-used-for-asynchronous-output-of-audit-trail-file*`]`#	
57	`[set pd_aud_async_buff_count = `*number-of-buffer-sectors-used-for-asynchronous-output-of-audit-trail-file*`]`#	
58	`[set pd_aud_async_buff_retry_intvl = `*retry-interval-for-allocation-of-a-buffer-to-be-used-for-asynchronous-output-of-audit-trail-file*`]`#	
59	`[set pd_aud_sql_source_size = `*size-of-SQL-statement-output-to-audit-trail*`]`	
60	`[set pd_aud_sql_data_size = `*size-of-SQL-data-output-to-audit-trail*`]`	
61	`[set pd_ha_acttype = monitor | server]`	System switchover facility
62	`[set pd_ha_unit = nouse]`	
63	`[set pd_ha_switch_timeout = Y | N]`#	
64	`[set pd_ha_server_process_standby = Y | N]`	
65	`[set pd_ha_agent = standbyunit | server | activeunits]`	
66	`[set pd_ha_max_act_guest_servers =`*maximum-number-of-acceptable-guest-BES*`]`	
67	`[set pd_ha_max_server_process =`*maximum-number-of-user-server-processes-inside-accepting-unit*`]`	
68	`[set pd_ha_process_count =`*number-of-processes-resident-inside-unit-after-acceptance-of-guest-BES*`]`	
69	`[set pd_ha_resource_act_wait_time = `*maximum-wait-time-for-resource-activation*`]`#	
70	`[set pd_ha_ipaddr_inherit = Y | N]`#	
71	`[set pd_rpl_hdepath = `*extracted-side-HiRDB-Datareplicator-directory-name*`]`	HiRDB Datareplicator
72	`[set pd_service_port = `*scheduler-process-port-number*`]`#	Communication processing

No.	Format	Operand category
73	[set pd_change_clt_ipaddr = 0 | 1]#	
74	[set pd_registered_port = "*port-number-reservation-range*" *[,"port-number-reservation-range"]...*]*	
75	[set pd_registered_port_check = Y | N | C | W]#	
76	[set pd_registered_port_level = 0 | 1]#	
77	[set pd_ipc_send_retrycount = *process-to-process-send-retries-count*]#	
78	[set pd_ipc_send_retrysleeptime = *process-to-process-send-retry-sleep-time*]#	
79	[set pd_ipc_send_count = *server-to-server-send-retries-count*]#	
80	[set pd_ipc_recv_count = *server-to-server-receive-retries-count*]#	
81	[set pd_inet_unix_bufmode = os | conf]#	
82	[set pd_ipc_inet_bufsize = *send-receive-buffer-size-for-server-to-server-communication*]#	
83	[set pd_tcp_inet_bufsize = *send-receive-buffer-size-for-communication-with-HiRDB-client*]#	
84	[set pd_listen_socket_bufset = 0 | 1]#	
85	[set pd_inet_unix_utl_bufmode = auto | conf]#	
86	[set pd_java_archive_directory = *"JAR-file-storage-directory"*]#	Java
87	[set pd_java_classpath = *"Java-class-path"*]#	
88	[set pd_java_runtimepath = *"Java-Runtime-Environment-root-directory"*]#	
89	[set pd_java_libpath = *"Java-virtual-machine-library-directory"*]#	
90	[set pd_java_stdout_file = *"Java-virtual-machine-standard-output-and-standard-error-output-destination-file"*]#	
91	[set pd_c_library_directory = "*C-library-file-storage-directory*"]	External C stored routines
92	[set pd_tmp_directory = *work-file-output-directory*]	Modifying the directory for work file output
93	**[putenv SHMMAX *maximum-shared-memory-segment-size*]**#	Shared memory

#: When this operand is omitted, the value specified for the same operand in the system common definition is used.

3.2 Operand explanations

3.2.1 Operands related to system structure

1) pd_unit_id = *unit-identifier*

~<identifier>((4 characters))

This operand is required.

Specifies an identifier for the unit. The unit identifier specified here must have been specified in the `pdunit -u` operand in the system common definition.

If multiple HiRDB/Single Servers are used, specify a unique identifier for each system.

For a HiRDB/Parallel Server, the specified identifier must be unique within the system.

Notes

When a unit identifier is changed, the following files must be reinitialized:

- Unit status files
- System log files

2) pd_hostname = *host-name*

~<host name>((1-32 characters))

Specifies the standard host name for the server machine of the primary system in which the unit was defined. For the host name, specify a host name, an IP address, or an FQDN.

You can specify the host name specified in the `-x` option of the `pdunit` operand in the system common definition. When specifying this host name, while you can specify a host name, an IP address, or an FQDN, be aware that the specification format must be the same as that used in the `-x` option of the `pdunit` operand. For example, if you specified an IP address in the `-x` option of the `pdunit` operand, you must also specify the IP address in this operand.

Specification guidelines

- Because this operand is closely related to the `pdunit` and `pdstart` operands in the system common definition, read the description of the `pdunit` and `pdstart` operands before you specify this operand.
- Specify the host name that is displayed when the `hostname` command is executed at the command prompt.
- Host names are case-sensitive.

Notes

If the `-n` option is specified in the `pdstart` operand to use the multi-connection address facility, this operand cannot be omitted. In this case, specify the standard host name in this operand.

3.2.2 Operands related to maximum concurrent executions

3) pd_max_server_process = *maximum-number-of-concurrently-activated-server-processes*

~<unsigned integer>((50-10000))

Specifies the maximum number of processes that can be activated at the same time in the unit. The number of server processes includes the number of processes for the system server, individual servers, utilities, and the like (the system server is a server that is used internally by HiRDB).

Specification guidelines

- Normally, omit this operand. When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the value of the following equation is assumed. Note that HiRDB automatically re-computes the value of this operand whenever you change the specification of any of the operands mentioned in the explanations below of the variables.

  Default value = $a + b \times (c + 27) + 73 + i + j + k + n + o$

  However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default for this operand is `100`.

- If you choose to specify this operand, use the following formula as a reference for determining the operand value. For a HiRDB/Parallel Server, determine values for all units, and use the largest of these values as a guideline.

  Recommended value = $a + b \times (c + d \times e + f) + d \times g + h + i + j + k + n + o + A$

Variable	Explanation of variable
a	**For HiRDB/Single Server:** *Value of pd_max_users* **For HiRDB/Parallel Server:** Total the values obtained using the following formulas for individual servers inside the unit. Back-end server: *Value of pd_max_bes_process* Dictionary server: *Value of pd_max_dic_process* For front-end servers: *Value of pd_max_users* • If the `pd_max_bes_process` or `pd_max_dic_process` operand is omitted, use the value of the `pd_max_users` operand. • If a unit contains multiple back-end servers, compute a value for each back-end server. • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, substitute the value of the `pd_ha_max_server_process` operand.
b	**For HiRDB/Single Server:** 1 **For HiRDB/Parallel Server:** Server count inside the unit (server count allocated to the unit by the `pdstart` operand of the system common definition) • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, add the value of the `pd_ha_max_act_guest_servers` operand (or the default, if this operand has been omitted).
c	Concurrently executable process count in full recovery processing (value of the `pd_max_recover_process` operand)
d	Maximum number of concurrently executable utilities (`2` is the default value for this operand)
e	Number of processes to be started in each server by the utility (set this variable to 10)
f	Number of processes to be started by HiRDB for controlling the server (set this variable to 6)
g	Number of processes to be started in each unit by the utility (set this variable to 10)
h	Number of processes to be started by HiRDB for controlling the unit (set this variable to 50)
i	**For HiRDB/Single Server:** 1 **For HiRDB/Parallel Server:** Number of back-end servers in the unit • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, substitute the value of the `pd_ha_max_act_guest_servers` operand.
j	**For HiRDB/Single Server:** *Value of pd_max_ard_process operand* **For HiRDB/Parallel Server:** *Number of back-end servers in the unit* $\times$ *pd_max_ard_process operand value + number of dictionary servers in the unit* $\times$ *pd_max_ard_process operand value* • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, add the following value: *Maximum value of pd_max_ard_process operand for a guest BES* $\times$ *pd_ha_max_act_guest_servers operand value*
k	Depends on the value specified for the `pd_process_terminator` operand:

Variable	Explanation of variable
	• `resident`: 1 • `fixed` (default value): `pd_process_terminator_max` `operand` value • `nonresident`: 0
n	**For HiRDB/Single Server:** `pd_dfw_awt_process` operand value **For HiRDB/Parallel Server:** *Number of back-end servers* × *pd_dfw_awt_process operand value* + *number of dictionary servers* × *pd_dfw_awt_process operand value* • For a unit to which the standby-less system switchover (effects distributed) facility is to be applied, add the following value: *Maximum value of the pd_dfw_awt_process operand for a guest BES* × *pd_ha_max_act_guest_servers operand value*
o	When using a memory database, substitute the following value: *p* + 128 + *XDS count in the unit* × 5 + 2 **If the unit has a system manager** *p*: 64 × *XDS count for system as a whole* **If the unit does not have a system manager** *p*: 0
A	Maximum number of transaction recovery processes that can be started $\downarrow$ *pd_trn_rcvmsg_store_buflen operand value* ÷ 72 $\downarrow$

Relationships to other operands

This operand's value limits the maximum value of the `pd_process_count` operand.

Notes

- This specification value includes the number of processes for servers and utilities in the unit. If this value is too small, the following might occur:
  - Unit or server startup process results in an error
  - Transaction recovery cannot be performed
  - HiRDB planned termination cannot be performed
- Because the number of processes that can actually be activated depends on factors such as the system resources, it might be necessary to adjust the resources or to change the locations of servers in some cases.

Effects on individual estimation formulas

If the value of the `pd_max_server_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Estimating desktop heap settings*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

3.2.3 Operands related to HiRDB processing

4) pd_server_entry_queue = spnfifo | fifo | loop

If contention occurs in the HiRDB server process during concurrent execution of UAPs, processing requests might sometimes be temporarily queued. This operand specifies what HiRDB must do in this case. Note that *process contention* in this case means that multiple processes are simultaneously trying to lock internal resources, such as tables and RDAREAs, when transactions are running on the HiRDB server process. Only a single process is allowed to lock internal resources at any point in time. *Spin* referred to in the following explanation means a process for acquiring the right to execute a lock. When another process releases the right to execute a lock, a process that is spinning has a higher probability of acquiring the right to execute a lock.

`spnfifo`:

A processing request that occurs first is given higher priority. However, because the process is spun a certain number of times before being registered in a queue, the priority order is not perfect. This method is used in version 06-00 and earlier.

`fifo`:

A processing request that occurs first is given higher priority than when `spnfifo` is specified. Because the process is not spun a certain number of times before a process is registered in a queue, the priority order is maintained better than when `spnfifo` is specified. This method also reduces the CPU load.

`loop`:

All processing requests are given the same priority. When processes are registered in a queue, they are spun at high speed. Specifying `loop` might improve the response during concurrent execution of UAPs. However, this method places a greater load on the CPU than other methods.

Specification guidelines

Normally, you need not specify this operand.

Change the specification value if the processing performance during concurrent execution of UAPs does not improve. Doing so might improve the performance.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `spnfifo`.

5) pd_thdlock_wakeup_lock = Y | N

Specifies a thread lock release notification method. Specify `Y` in this operand to ensure that release notifications are transmitted.

`Y`:

When issuing a thread lock release notification, a new separate lock is temporarily obtained.

`N`:

When issuing a thread lock release notification, no new separate lock is temporarily obtained.

This is the HiRDB processing mode for versions 06-02 and earlier.

Specification guidelines

Specify `Y` in this operand.

Notes

Note the following about specifying `N` or omitting this operand:

- Transactions might occur that have longer execution times than for other transactions, reducing response during multiplexed UAP execution.
- When no notification is sent that a thread lock has been released, there might be a delay equal to the amount of time specified in `pd_thdlock_pipe_retry_interval` in obtaining the lock that is waiting for release.

6) pd_thdlock_pipe_retry_interval = *thread-lock-release-check-interval*

~<unsigned integer>((0-2147483647))(microseconds)

Specifies in microseconds the interval at which to check for thread lock release.

Specification guidelines

If the value that is set is the same as or greater than the default value and all the following conditions are satisfied, CPU usage might decrease:

- `pd_thdlock_wakeup_lock` = `Y` is specified.
- Processing performance does not increase during multiplexed UAP execution.
- The CPU usage rate is very high.

However, transactions tend to occur that have longer execution times than for other transactions.

Do not specify this operand when the conditions above are not applicable.

Note

If a value less than the default value is specified, release checking is repeated at short intervals, which might increase the CPU usage rate.

7) pd_thdlock_retry_time = thread-lock-sleep-time

~<unsigned integer>((1-1000000))(microseconds)

Specifies in microseconds the thread lock sleep time. If this operand is specified when all the conditions listed below are satisfied, the CPU usage rate might decrease.

- The CPU usage rate is very high.
- Reducing the CPU usage rate is necessary, even if it results in a reduction in performance.
- `0` is specified for the `pd_thdlock_sleep_func` operand.

Do not specify this operand when the conditions above are not applicable. The following describes the HiRDB processing based on the combination of the `pd_thdlock_sleep_func` and the `pd_thdlock_retry_time` operand values:

pd_thdlock_sleep_func operand value	pd_thdlock_retry_time operand value	
	1 to 10000	10001 to 1000000
0	Each process stands by for the thread lock sleep time specified by `select()` or `Sleep()`.	
1	The OS determines process allocation using `sched_yield()` or `SwitchToThread()` (the `pd_thdlock_retry_time` operand value is ignored).#	Each process stands by for the thread lock sleep time specified by `select()` or `Sleep()`.

#: Because processes do not go on standby, CPU usage increases.

Specification guidelines

If you choose to specify this operand, start by specifying `10000`. If the CPU usage rate is still too high, increase the value.

Notes

- Reducing the value might not change the performance.
- Specifying `1000` or a greater value might have an adverse effect on performance.

Relationship to other operands

- If `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default value for this operand is `10000`.

8) pd_thdspnlk_spn_count = ***thread-spin-lock-spin-count***

~<unsigned integer>((0-2147483647))

Specifies a spin count for thread spin lock. Specifying this operand when all of the following conditions are satisfied might improve the system performance. Otherwise, there is no need to specify this operand.

- An ample margin exists in the CPU usage rate.
- You want to improve performance, even if doing so increases the CPU usage rate.

Specification guidelines

- If this operand is to be specified, specify a value that is greater than `512`.
- Because the specification value depends on the OS type, the processor type, the machine performance, the disk performance, the UAP content, and the number of UAPs concurrently being executed, there is no

clear guideline. Determine an appropriate value by varying the specification value and measuring the performance.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `512`.

Notes

- If the value of this operand is too large, the CPU usage rate might increase, causing problems such as slower OS operation. In this case, decrease the operand value.
- Increasing the value of this operand might not always improve performance.

9) pd_db_io_error_action = dbhold | unitdown

Specifies the processing to be performed by HiRDB when an input/output error occurs in an RDAREA (excluding the master directory RDAREA). If an input/output error occurs in the master directory RDAREA, HiRDB (a unit for a HiRDB/Parallel Server) always terminates abnormally regardless of the specification in this operand. For the actions to be taken when an RDAREA input/output error occurs, see the *HiRDB Version 9 System Operation Guide*.

An input/output error in this case refers to an error that occurs when a file manipulation attempt by HiRDB fails due to a cause that cannot be determined by HiRDB. When such an error occurs, `-1544` is output as the error code returned in response to a HiRDB file system access request.

`dbhold`:

When an input/output error occurs in an RDAREA, the RDAREA is placed in an error shutdown state.

`unitdown`:

If an input/output error occurs in an RDAREA, HiRDB (a unit for a HiRDB/Parallel Server) terminates abnormally. However, if an input/output error occurs again following abnormal termination, the RDAREA is placed in an error shutdown state. To enable the specification of `unitdown` again, take one of the following actions:

- Start HiRDB normally.
- Execute the system reconfiguration command (`pdchgconf` command).

Specification guidelines

To determine the specification value for this operand, see *HiRDB processing when an RDAREA I/O error occurs* in the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `dbhold`.

Notes

- HiRDB terminates abnormally if an input/output error occurs while `unitdown` is specified. Consequently, in the following cases, the processing target RDAREA might go onto error shutdown status:

  ● The UAP or utility is executing in the pre-update log acquisition mode or the no-log mode

  ● The UAP or utility is being executed on a user LOB RDAREA that has been placed in the no-log mode by specification of `NO` in the `RECOVERY` operand of `CREATE TABLE`.

  If you use the facility for taking a unit down when a physical error is detected, avoid running these operations, if possible. If you need to run these operations, make a backup prior to running the UAP or utility in case recovery from an RDAREA error shutdown has to be performed. For details about making back-ups, see the *HiRDB Version 9 System Operation Guide*.
- If an input/output error occurs during the startup or termination process, HiRDB does not terminate abnormally, even if `unitdown` is specified.
- During recovery processing by the database recovery utility (`pdrstr`), HiRDB does not terminate abnormally even though `unitdown` is specified. In this case, re-execute `pdrstr` to perform recovery.

Relationship to other operands

This operand is related to the following operands:

- `pd_mode_conf` operand

- `pd_db_access_error_action` operand
- `pd_db_hold_action` operand

If `unitdown` is specified in more than one of the `pd_db_io_error_action`, `pd_db_access_error_action`, and `pd_db_hold_action` operands, the operand value that takes effect is determined in the following order:

1. `pd_db_io_error_action` operand
2. `pd_db_access_error_action` operand
3. `pd_db_hold_action` operand

If more than one RDAREA input/output, file access, or physical error has occurred, determine the error that caused unitdown based on the above priority. In addition, see the message that is issued.

3.2.4 Operands related to full recovery processing

10) pd_max_recover_process = ***concurrently-executable-full-recovery-processes-count***

~<unsigned integer>((1-10))

Specifies the number of processes to be recovered (`REDO` processes) during full recovery processing. For a HiRDB/Parallel Server, this operand specifies the number of processes to be recovered (`REDO` processes) per server (dictionary server or back-end server).

Condition

You need to use the raw I/O facility.

Specification guidelines

- There are three or more logical volumes for which an RDAREA is defined (per server): 3
- There are fewer than three logical volumes for which an RDAREA is defined (per server): Number of logical volumes
- Increasing the value of this operand increases the input/output concurrency during full recovery processing, and thus can shorten the recovery time. However, because a number of processes equaling *value of this operand* × *server count* are started, determine a value by taking the aforementioned specification value guideline and HiRDB resources into consideration.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `3`.

3.2.5 Operands related to transaction decision processing

11) pd_trn_rcvmsg_store_buflen = ***transaction-recovery-message-queue-size***

~<unsigned integer> ((4096-25600000))(bytes)

When HiRDB performs transaction recovery processing, it registers the transaction to be recovered in the transaction recovery message queue. This operand specifies the transaction recovery message queue size.

Specification guidelines

- Normally, there is no need to specify this operand.
- If the `KFPS00854-W` message (`server=_trnrcv`) is issued during HiRDB operation, consider specifying this operand. For the formula for estimating the specification value, see *pd_trn_rcvmsg_store_buflen* operand in the system common definition.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `4096` is assumed.

Notes

- If a value greater than 4,096 is specified in this operand, the shared memory size for unit controller increases. For details, see *Formulas for shared memory used by a unit controller* in the *HiRDB Version 9 Installation and Design Guide*.
- If a value greater than 4,096 is specified in this operand, the maximum number of transaction recovery processes (`pdtrnrvd`) increases. For details about the formula for determining the maximum number of `pdtrnrvd`s, see *pd_trn_rcvmsg_store_buflen* operand in the system common definition.

Relationship to other operands

This operand is related to the `pd_max_server_process` operand.

3.2.6 Operands related to system monitoring

12) pd_watch_time = *maximum-response-wait-time*

~<unsigned integer>((0, 600-65535)) (seconds)

This operand is provided to ensure compatibility with earlier versions, so it is not necessary to specify it. This operand is applicable only to HiRDB/Parallel Server.

Specifies the maximum amount of time a HiRDB server process is to wait for a response from another HiRDB server process (dictionary server or back-end server). When the set maximum response wait time is exceeded, the HiRDB server process suspends processing (although some HiRDB server processes might not be suspended).

Advantage

If the HiRDB server does not halt execution of an SQL statement, command, or utility, even though the HiRDB client has canceled the SQL statement, command, or utility execution (by forced termination, for example), resources might remain locked for a long time. Specifying this operand can sometimes reduce the period of time this lock is in effect.

Specification guidelines

Specify the largest value among the following time values:

- Time specified by the `PDCWAITTIME` operand of the client environment definition
- Time specified by the `PDLCKWAITTIME` operand of the client environment definition
- Time specified by the `pd_lck_wait_timeout` operand
- Processing time of the SQL statement, command, or utility with the longest execution time

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

Notes

- If a value in the range `1-599` is specified in this operand, the specified value will be rounded up to `600`.
- When the SQL execution time is set as the maximum response wait time, we recommend that you set this operand to `0` (or that the operand be omitted) and that the `PDCWAITTIME` operand of the client environment definition be specified. For details about the `PDCWAITTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.
- If a value smaller than the value set in `PDCWAITTIME` is specified for this operand, additional time might be required for transaction recovery when `pd_watch_time` is exceeded. If this occurs, an error will likely result and locked resources will not be released in a timely manner.

13) pd_down_watch_proc = *upper-limit-for-server-process-abnormal-terminations*[*, monitoring-interval*]

This operand is used for monitoring the number of abnormal terminations of a HiRDB server process. Processes to be monitored are those that are abnormally terminated by `PDCWAITTIME` over or aborting.

If abnormal terminations of server processes occur frequently, new services might not be accepted. However, because server process abnormal termination does not cause HiRDB abnormal termination, HiRDB is in an online stopped state in effect. When this operand is specified, you can pull HiRDB out of this state by restarting it.

***upper-limit-for-server-process-abnormal-termination*: ~<unsigned integer>((0-65535))**

If abnormal terminations of server processes exceed the value specified in this operand, HiRDB (an applicable unit for a HiRDB/Parallel Server) is abnormally terminated. This is called the facility for monitoring abnormal process terminations. For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

For a HiRDB/Single Server, abnormal terminations of single server processes are counted. For a HiRDB/ Parallel Server, the total of the abnormal terminations in the front-end servers, back-end servers, and dictionary servers inside the unit is counted.

If `0` is specified, abnormal terminations of server processes are not monitored.

***monitoring-interval*:~<unsigned integer>((10-3600)) (seconds)**

Specifies the interval (in seconds) for monitoring abnormal terminations of server processes.

For example, if `100` is specified, abnormal terminations of server processes are monitored every 100 seconds.

Advantages

- Restart of HiRDB refreshes memory and resource statuses, improving the processing efficiency.
- If abnormal termination of server processes occurs frequently, HiRDB is abnormally terminated, and thus the system can be switched over immediately.

Notes

- Do not use this operand in a system that does not allow abnormal termination.
- When a server process is abnormally terminated, the `KFPS01820-E` message is output. Although this message is also output when the server process is abnormally terminated by the `pdcancel` command, this is not counted as an abnormal termination.
- For a mutual system switchover configuration, multiple HiRDBs are activated on the same server machine when system switchover occurs. As a result, the system traffic might increase, causing an adverse effect instead. Therefore, if you specify this operand, we recommend that you restart HiRDB in the system that terminated abnormally.
- If a HiRDB server process terminates abnormally repeatedly, a large amount of troubleshooting information will be output, resulting in frequent input to and output from the HiRDB operating directory, possibly leading to a full disk.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the following values are assumed:

- Maximum number of process abnormal terminations: `0`
- Monitoring interval: `600`

Remarks

- If HiRDB is abnormally terminated by the facility for monitoring abnormal process terminations, the `KFPS01821-E` and `KFPS00729-E` messages are output.
- The following table shows the causes of server process abnormal termination and the server processes that are included in the abnormal termination count.

Cause of server process abnormal termination	Inclusion in abnormal termination count			
	Single server process	Front-end server process	Dictionary server process	Back-end server process
`PDCWAITTIME` operand value of the client environment definition has been exceeded.	Y	Y	N#1	N#1
`pdcancel` command	N	N#2	N	N
Internal forced termination (HiRDB internally issues `SIGKILL` and terminates a process)	Y#3	Y#3	N#1	N#1
Abort	Y	Y	Y	Y
One of the following:	Y	Y	N	N

Cause of server process abnormal termination	Inclusion in abnormal termination count			
	Single server process	Front-end server process	Dictionary server process	Back-end server process
• Abnormal termination of server process by transaction recovery processing in an OLTP system Abnormal termination of server process by XDS transaction recovery processing[#4]				
Abnormal termination of process other than those described here	Y	Y	Y	Y

Legend:

Y: Included in abnormal termination count

N: Not included in abnormal termination count

#1

If an error is detected in a transaction branch, the abnormal terminations of the front-end server process that have occurred in the same transaction branch are counted.

#2

If the `pdcancel` command is used to forcibly terminate a back-end server process or dictionary server process, the front-end server process is internally and forcibly terminated. In this case, the abnormal termination of the front-end server process might be counted in some cases.

#3

If an error is detected in a global transaction by an OLTP system, the abnormal terminations of the single server process or front-end server process that has occurred in the same global transaction are counted.

#4

If the completion status of a transaction executed from XDS on a server that provides the primary functionality cannot be determined, XDS transaction recovery processing might produce a rollback and the server process might terminate abnormally.

3.2.7 Operands related to SQL runtime warning output facility

14) pd_cwaittime_report_dir = *SQL-runtime-warning-information-file-output-destination-directory*

~<path name>

Specifies an absolute path name as the output destination directory for the SQL runtime warning information file. Two SQL runtime warning information files (`pdcwwrn1` and `pdcwwrn2`) are created under the directory specified here.

If this operand is omitted, no SQL runtime warning information file is output. However, the warning message (`KFPA20009-W`) is still output.

For a HiRDB/Parallel Server, the SQL runtime warning information files are output to the server machine containing the front-end server to which the UAP that issued the warning target SQL statement is connected.

Operand rules

- Specify no more than 255 characters for the path name.
- The path name is not case sensitive.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. If the same operand is also omitted from the system common definition, no SQL runtime warning information files are output. However, a warning message (`KFPA20009-W`) is issued.

15) pd_cwaittime_report_size = *SQL-runtime-warning-information-file-maximum-size*

~<unsigned integer>((2048-2147473627)) (bytes)

Specifies the maximum size for the SQL runtime warning information file. The value specified in this operand indicates the size of a single SQL runtime warning information file. Therefore, be careful about the value you specify for this operand because two SQL runtime warning information files are created. For example, if you specify `10,000`, two files, each with a maximum size of 10,000 bytes, are created under the directory.

Specification guidelines

Use the following formula as a guideline when determining the value to be specified for this operand.

(1,280 + *SQL statement size* (bytes)) × *number of pieces of warning information to be stored in file*

If a comment or SQL optimization is specified for the SQL statement, also include the size of the comment and the specified SQL optimization size (bytes) in the SQL statement size.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `100000`.

Remarks

- If the volume of data that is output to an SQL runtime warning information file exceeds the value specified by this operand, the output destination is switched to the other file. The two files are alternately used as this process is repeated. During this process, the oldest information is deleted from the switching destination file.
- If the volume of the SQL runtime warning information that is output at one time exceeds the file size, not all of the SQL runtime warning information is output. Only the information that fits in the file size is output. In this case, the hash mark (`#`) is added to the end of the SQL runtime warning information.

3.2.8 Operands related to the facility for output of extended SQL error information

16) pd_uap_exerror_log_dir = *SQL-error-report-file-storage-directory*

~<path name of up to 255 characters>

Specifies an absolute path name for the directory in which to store SQL error report files.

Two SQL error report files are created in the specified directory. Their file names are `pduaperrlog1` and `pduaperrlog2`.

For details about the facility for output of extended SQL error information, see the *HiRDB Version 9 UAP Development Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. If the same operand is also omitted from the system common definition, no SQL error information is output to the SQL error report files.

Note

The specified path name is not treated as being case sensitive.

17) pd_uap_exerror_log_size = *SQL-error-report-file-maximum-size*

~<unsigned integer>((2048-2147483647))(bytes)

Specifies the maximum size of an SQL error report file. The value specified by this operand applies to each of the two SQL error report files that are to be created. When the volume of data that is output to an SQL error report file exceeds the value specified by this operand, the output destination is switched to the other file. The two files are used alternately as this process is repeated. If the volume of the SQL error information that is output at one time exceeds the value specified by this operand, the first through the [*specified value* - 1]-th bytes (up to the 999,999th byte if `1000000` is specified for this operand) of SQL error information is output. In this case, the hash mark (`#`) is added to the end of the SQL error information.

Specification guidelines

Determine the value to be specified for this operand by taking into consideration the volume of SQL error information that you want to retain. You can use the following computation formula:

(*A* + *B*) × *volume to be retained*

- *A* = 1,100 + *SQL statement size* (bytes)

  This is the size of each piece of SQL error information, excluding the parameter information output size. If an SQL statement contains a comment or the description of SQL optimization specification, the size of the comment or SQL optimization specification must also be included in the SQL statement size. For details about comments and SQL optimization specification, see the manual *HiRDB Version 9 SQL Reference*.

- $B = (\uparrow$ *pd_uap_exerror_log_param_size-operand-value* $\div 16 \uparrow + 1) \times 89 \times$ *parameter-count*

  This is the parameter information output size.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `1000000`.

18) pd_uap_exerror_log_param_size = ***maximum-data-size-of-parameter-information-to-be-output-to-client-error-log-file-and-SQL-error-report-file***

~<unsigned integer>((0-32008))(bytes)

Specifies the maximum data size for the parameter information to be output to a client error log file and an SQL error report file.

- When `1` or a value greater than `1` is specified

  Parameter information is output to a client error log file and an SQL error report file.

- When `0` is specified

  Parameter information is not output to a client error log file or an SQL error report file.

If the parameter information is in the variable-length character string type, BLOB type, or BINARY type, the data size area also is included in the specified value.

If the size of the parameter information to be output to a client error log file and an SQL error report file exceeds the value specified for this operand, only the parameter information that fits in the file size is output, and the remainder is discarded.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

Relationship to other operands

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDUAPEXERLOGPRMSZ` operand in the client environment definition. If both this operand and the `PDUAPEXERLOGPRMSZ` operand are specified, the `PDUAPEXERLOGPRMSZ` operand in the client environment definition takes precedence.

For details about the `PDUAPEXERLOGPRMSZ` operand, see the *HiRDB Version 9 UAP Development Guide*.

3.2.9 Operands related to lock

19) pd_lck_wait_timeout = ***lock-release-wait-time***

~<unsigned integer>((0-65535)) (seconds)

Specifies in seconds the maximum amount of time to wait for lock release (the maximum amount of time in lock release wait status).This is the elapsed time from when a lock release request is placed in wait status until it is released.

If the wait status is not released within the specified amount of time, the SQL statement will return an error. When `0` is specified, lock wait time will not be monitored and waiting will continue until the wait status is released.

Specification guidelines

When you specify this operand, check that the following condition is satisfied:

Value specified in `pd_watch_time` operand

> value specified in `PDCWAITTIME` of client environment definition

> value specified in this operand

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is the larger of the following two values:

- `180`
- Value of the `pd_watch_time` operand

However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `180`.

Note

If this operand is omitted, the value of the `pd_watch_time` operand might be used as the default value in some cases. Although the `pd_watch_time` operand is invalid, even if it is specified for a HiRDB/Single Server, the specified value might be used as the default value for the `pd_lck_wait_timeout` operand in some cases. Therefore, we recommend that you omit the `pd_watch_time` operand for a HiRDB/Single Server.

Relationship to client environment definition

The value of this operand can be modified for a client. To do so, specify `PDLCKWAITTIME` in the client environment definition. For details about `PDLCKWAITTIME`, see the *HiRDB Version 9 UAP Development Guide*.

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_deadlock_check`
- `pd_lck_deadlock_check_interval`

20) pd_lck_release_detect = interval | pipe

Specifies the method to be used by HiRDB to detect lock release (method of detecting whether the process that locked a resource has released that lock).

`interval`:

Determine the lock release status by checking the lock management area at a regular interval.

`pipe`:

Use a pipe file (process-to-process communication pipe) to receive the lock release notice.

Specification guidelines

Following are guidelines for specifying this operand:

Specified value	HiRDB processing	Advantages and application criteria
`interval`	Determines whether the lock has been released by checking the lock management area in the shared memory. HiRDB checks the lock management area at a regular interval; this interval is specified by the `pd_lck_release_detect_interval` operand.	Even if the process that has locked a resource releases the lock, the release will not be detected until the next time the lock management area is checked. Consequently, a UAP that has a fast processing time per transaction might end up waiting for a long time. However, this wait does not place any load on the CPU or open any file. If a small value is specified for the `pd_lck_release_detect_interval` operand, the CPU usage might increase too much, adversely affecting the throughput. Specifying `interval` has the effect of reducing the CPU load when a slow CPU is being used.
`pipe`	Uses a semaphore to determine whether the lock has been released. The process that has locked a resource sends a lock release notice to the process that is waiting for release. When the process that has locked the resource releases it, the process that is waiting for the release can detect the release	• Throughput improves if the processing time per transaction does not exceed the value specified by the `pd_lck_release_detect_interval` operand. • CPU workload increases if lock-release waits occur frequently. • If a high-speed CPU is used, this method improves throughput.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, `pipe` is assumed for a HiRDB/Single Server and `interval` for a HiRDB/Parallel Server.

21) pd_lck_release_detect_interval = *lock-release-detection-interval*

~<unsigned integer>((1-1000)) (milliseconds)

Specifies the interval at which the lock management area is to be checked.

- When 49 or less is specified
  The interval will begin at the value specified by this operand and thereafter will be 50 milliseconds.
- When 50 or more is specified
  The interval will begin at 50 milliseconds and thereafter will be the value specified by this operand.

Condition

The `interval` option must be specified for the `pd_lck_release_detect` operand.

For a HiRDB/Parallel Server, `interval` is the default value.

Specification guidelines

- As a rule, this operand can be omitted.
  Consider modifying the specified value if transaction processing performance declines significantly because many locks occur in a short period of time and a lock release wait has occurred.
- If too small a value is specified, the CPU workload will increase if lock release waits occur frequently.
- If too large a value is specified, the wait time might increase.
- The value to be specified can be determined by referring to `WAIT TIME` in the statistical information related to system operation from the statistics analysis utility. If the wait time that is output in the statistical information is smaller than the value of this operand, reduce the value of this operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `10`.

22) pd_lck_deadlock_info = Y | N

Specifies whether deadlock information, timeout information and locked resource management table information are to be output. These types of information are output to a directory named `%PDDIR%\spool\pdlckinf`. For details about deadlock, timeout, and locked resource management table information, see the *HiRDB Version 9 System Operation Guide.*

`Y`:

Outputs deadlock information, timeout information, and locked resource management table information.

`N`:

Does not output deadlock information, timeout information, or locked resource management table information.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `Y`.

Relationship to other operands

- When `N` is specified in the `pd_lck_deadlock_check` operand, this operand is assumed to be `N`.
- This operand is related to the `pd_lck_deadlock_check` operand.

23) pd_lck_deadlock_check = Y | N

Specifies whether checking for deadlock is to be performed.

`Y`: Checks for deadlocks.

`N`: Does not check for deadlocks.

Specification guidelines

In transaction systems that do not generate deadlocks, specifying `N` in this operand can improve SQL execution performance. This is especially true when the interval check mode is used as the deadlock detection method, because lock performance can deteriorate whenever a deadlock is detected if the number of pool partitions for locks is increased. Consequently, we recommend that you specify `N` in this operand in the case of designing a transaction system that does not generate deadlocks.

In a transaction system that does have potential to generate deadlocks, specify `Y` in this operand. Specifying `N` would mean that when a deadlock did occur, SQL would not terminate until the time specified in the `pd_lck_timeout` operand has elapsed. Also, because HiRDB does not output deadlock information, it might become impossible to determine what caused the deadlock.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `Y`.

Notes

When you specify `N` in this operand, a transaction that generates a deadlock will not result in an error. Instead, the transaction will be canceled on the basis of one of the following:

- The maximum time for checking lock release wait time elapses and the SQL statement returns an error.
- The maximum wait time of the HiRDB client elapses and the request returns an error to the UAP.

Relationship to client environment definition

This operand is related to the following client environment definitions:

- `PDCWAITTIME`
- `PDLCKWAITTIME`

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_deadlock_info`
- `pd_lck_pool_partition`
- `pd_fes_lck_pool_partition`
- `pd_lck_deadlock_check_interval`
- `pd_lck_wait_timeout`

24) pd_lck_deadlock_check_interval = *deadlock-check-interval*
~<unsigned integer>((1-2000000000))(msec)

Specifies the interval for performing checking during monitoring for occurrence of deadlocks in the interval check mode.

Condition

Conditions for this operand depend on the server.

For a single server, back-end server, or dictionary server, both the following conditions must be satisfied:

- `Y` is specified in the `pd_lck_deadlock_check` operand
- `2` or greater is specified in the `pd_lck_pool_partition` operand

For a front-end server, both the following conditions must be satisfied:

- `Y` is specified in the `pd_lck_deadlock_check` operand.
- `2` or greater is specified in the `pd_fes_lck_pool_partition` operand

Specification guidelines

Specify a small value in this operand to reduce the time between occurrence and detection of a deadlock.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `1000`.

Notes

- If too small a value is specified in this operand, system performance might be degraded.
- If too large a value is specified in this operand, a transaction might be canceled for the following reasons:
  - The maximum time for checking the lock release wait time elapses and the SQL statement returns an error.
  - The maximum wait time of the HiRDB client elapses and the request returns an error to the UAP.

Relationship to client environment definition

This operand is related to the following client environment definition:

- `PDCWAITTIME`

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_deadlock_check`
- `pd_lck_pool_partition`
- `pd_fes_lck_pool_partition`
- `pd_lck_wait_timeout`

3.2.10 Operands related to buffers

25) pd_thread_stack_expand_size = ***stack-extension-size-per-thread***

~<unsigned integer> ((0-2146435072))(bytes)

This operand increases the stack size per thread for the threads controlled by HiRDB.

Normally, omit this operand. Change the operand value only if asked to by a maintenance engineer.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `0` is assumed.

3.2.11 Operands related to shared memory

26) pd_shmpool_attribute = free | fixed

Specifies whether to fix the shared memory used by the HiRDB unit controller in memory. Note that whether this feature is supported depends on the version of Windows you are using. For details about the prerequisites for Windows, see *Page locking shared memory* under *System design* in the *HiRDB Version 9 Installation and Design Guide*. If this feature is not supported, this operand is ignored even if `fixed` is specified.

`free`:

Do not fix the shared memory in memory. Depending on the size of the real memory, another page of shared memory might be created, adversely affecting performance.

`fixed`:

Fix the shared memory in memory.

Advantage

Fixing the shared memory to be used by the HiRDB in the memory ("fixed" specified) prevents shared memory paging, thus improving the access performance to the shared memory.

Specification guidelines

Determine whether to fix shared memory based on the computed shared memory size and the real memory size of the server machine. Because there is a limit to the amount of memory that can be fixed for the HiRDB, fixing too much shared memory might cause frequent paging of other types of memory. How much real memory there is and the amount of memory to be occupied by the shared memory pool need to be carefully evaluated.

For the formulas used to calculate the shared memory sizes to be used by the unit controller and individual servers, see the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `free`.

Notes

The following notes apply to specification of `fixed` in a Windows system that supports page fixing:

- The shared memory must be allocated to the paging file with the `pdntenv` command. For details about the `pdntenv` command, see the manual *HiRDB Version 9 Command Reference*.
- When you estimate the required memory size, note that the shared memory is acquired with its size rounded up to the page size of a Windows large page. To check the page size of a Windows large page, use the `pdntenv -os` command.

27) pd_dbbuff_attribute = free | fixed

Specifies whether to fix the shared memory used by the global buffers in the real memory. Note that whether this feature is supported depends on the version of Windows you are using. For details about the prerequisites for Windows, see *Page locking shared memory* under *System design* in the *HiRDB Version 9 Installation and Design Guide*. If this feature is not supported, this operand is ignored even if `fixed` is specified.

`free`:

Do not fix the shared memory to be used by the global buffer in the real memory.

`fixed`:

Fix the shared memory to be used by the global buffer in the real memory.

Advantage

Fixing the shared memory to be used by the global buffer in the real memory (`fixed` specified) prevents shared memory paging, thus improving the performance of accesses to the shared memory.

Specification guidelines

- To emphasize performance when there is ample real memory, specify `fixed`.
- Determine whether to fix shared memory based on the computed shared memory size and the real memory size of the server machine. If a memory page that is quite large relative to the real memory is fixed, the result might be frequent paging or a virtual memory shortage. Therefore, how much real memory there is and the amount of memory to be occupied by the shared memory pool need to be evaluated carefully. For the formulas used to calculate the shared memory size to be used by the global buffer, see the *HiRDB Version 9 Installation and Design Guide*.
- When the size of the shared memory to be page-fixed is subtracted from the size of the real memory, as a guideline, make sure that the result does not equal or exceed half the size that is obtained by subtracting the size of the shared memory to be page-fixed from the swap area size.

Relationship to other facilities

- If `fixed` is specified, the shared memory used by a dynamically modified global buffer is also fixed in the real memory. Therefore, carefully consider the real memory size before adding or modifying a global buffer.

Notes

The following notes apply to specification of `fixed` in a Windows system that supports page fixing:

- The shared memory must be allocated to the paging file with the `pdntenv` command. For details about the `pdntenv` command, see the manual *HiRDB Version 9 Command Reference*.
- When you estimate the required memory size, note that the shared memory is acquired with its size rounded up to the page size of a Windows large page. To check the page size of a Windows large page, use the `pdntenv -os` command.
- In the `SHMMAX` operand, specify a value that is rounded up to the page size of a Windows large page.
- If the shared memory for the global buffer pool cannot be fixed in memory when HiRDB starts, the `KFPH23045-W` message is issued. In this case, even if `fixed` is specified, HiRDB resumes processing without page-fixing some, or all, of the shared memory in memory.

3.2.12 Operands related to statistical information

28) pd_stj_file_size = *maximum-statistics-log-file-size*
~<unsigned integer>((64-1000000)) (kilobytes)

Specifies in kilobytes the maximum size of a statistics log file.

Two statistics log files are provided. When the amount of stored statistics log information reaches the size specified here, the statistics log files are swapped.

Specification guidelines

- For details about how to determine the value to be specified for this operand, see *C.1 Formulas for determining size of statistics log file (pd_stj_file_size)*.
- Specify this operand such that the following relationship with the `pd_stj_buff_size` operand is maintained:

`pd_stj_file_size ≥ pd_stj_buff_size × 2`

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `1024`.

29) pd_stj_buff_size = *statistics-log-buffer-size*

~<unsigned integer>((32-512)) (kilobytes)

Specifies the statistics log buffer size.

Specification guidelines

The default value of 32 is appropriate when the following types of statistical information are not to be output:

- SQL object execution information
- Statistics about SQL object transmission

If these types of statistical information are to be output, specify the value obtained by adding 32 to the result obtained from the following formula (however, if the result exceeds 512, specify 512):

$(a \div 1{,}024) \times (0.03 \div b)$

a: Statistics log output volume (bytes)

For details, see *C.1 Formulas for determining size of statistics log file (pd_stj_file_size)*.

b: Statistical information output time (seconds)

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `32`.

3.2.13 Operands related to RPC trace information

30) pd_rpc_trace = Y | N

Specifies whether RPC trace information is to be collected. HiRDB maintenance information is output in the RPC trace information.

Normally, omit this operand.

`Y`: Collect RPC trace information.

`N`: Do not collect RPC trace information.

Note

Specifying `Y` for this operand degrades communication performance.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `N`.

31) pd_rpc_trace_name = *"name-of-RPC-trace-collection-files"*

~<path name of up to 254 characters>

Specifies as an `absolute` path name the file name for the RPC trace files. Three RPC trace files are created, with 1, 2, and *l* suffixed to the specified file name.

Note

Files with a maximum size of `pd_rpc_trace_size value × 2` are created under the directory specified by this operand. Pay attention to the file size.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default value is as follows:

`%PDDIR%\spool\rpctr`

32) pd_rpc_trace_size = *RPC-trace-collection-file-size*

~<unsigned integer>((1024-2147483648)) (bytes)

Specifies the size of the RPC trace files in bytes.

The value specified by this operand determines the size of both RPC trace file 1 and RPC trace file 2.

Specification guidelines

An RPC trace collects in a single file in time order the information on the communications among all processes. If the volume of this information exceeds the specified value, trace data output is switched to the other file, which means that older trace information will be overwritten. When this happens, there might not be enough trace information available, making troubleshooting difficult. For this reason, specify at least `1000000` for this operand.

Note

The value specified by this operand does not apply to RPC trace file *l*, because *l*'s size is fixed at 0 bytes.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `4096`.

3.2.14 Operands related to performance trace information

33) pd_prf_level = 00000007 | 0000001f | 0000007f | 000001ff | 00000000

Specifies the performance trace information collection level in hexadecimal notation. For details about the information that can be collected according to the performance trace collection level, see *Details about performance trace information and collection points* in the *HiRDB Version 9 System Operation Guide*.

The following table describes the collection level determined by the value of the `pd_prf_level` operand and the information that is collected.

Table 3–1: Collection level determined by the value of the pd_prf_level operand and the information that is collected

pd_prf_level operand value	Collection level	Positioning	Information that is collected
`00000007` (Minimum level)	`00000004`	Provides the minimum amount of information.	Traces of `CONNECT` operation, utility execution, and specific events
`0000001f` (Default level)	`00000004`, `00000010`	Allows the boundaries with HiRDB (entry and exit) to be identified.	Traces mostly before and after communication with HiRDB in addition to the traces obtained at the minimum level
`0000007f` (Detail level)	`00000004`, `00000010`, `00000040`	Provides the information required for tracing SQL processing.	Traces that can be used to check the flow of HiRDB's internal processing in addition to the traces obtained at the default level
`000001ff` (Maintenance level)	`00000004`, `00000010`, `00000040`, `00000100`	Provides the maintenance information that can be used in the event of an error.	Traces that includes iteration processing such as internal locking in addition to the traces obtained at the detail level
`00000000` (Suppression level)	Not collected	Suppresses the collection of trace data.	Not collected

Specification guidelines

- We recommend that you specify `0000001f` (default level) so that when you investigate performance-related errors, you can quickly identify the cause of the problem. If a decrease in online performance is acceptable, such as during environment configuration, consider specifying `0000007f` (detail level).
- Normally there is no need to specify `000001ff` (maintenance level). Specify this level only if you are requested to do so by the support service.
- Do not specify `00000000` (suppression level) because, in the event of an error, this setting makes it difficult to determine its cause.

Notes

- If the number of performance trace collection points is increased and the collection level is changed to acquire more detailed performance traces, online performance might decrease.

- Do not specify a value other than `00000007`, `0000001f`, `0000007f`, `000001ff`, or `00000000`. If any other value is specified, it becomes more difficult to verify performance and to investigation errors.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `00000007` is assumed.

Relationship to other operands

This operand is related to the following operands:

- `pd_prf_file_count`
- `pd_prf_file_size`

Remarks

By using the `pdprflevel` command, you can change the collection level even while HiRDB is running.

34) pd_prf_file_count = *number-of-performance-trace-information-file-generations*

~<unsigned integer> ((3-256))

Specifies the number of performance trace information file generations that will be retained.

Specification guidelines

- Normally, this operand does not need to be specified.
- Consider specifying this operand if you want to retain the information in the performance trace information file for a long time. Estimate the number of performance trace information file generations and the file size from the following total capacity of the performance trace information file:

  Total capacity of the performance trace files = *pd_prf_file_count operand value* × *pd_prf_file_size operand value*

  Example

  This example specifies the total capacity of the performance trace information files required for retaining one hour worth of performance trace information.

pd_prf_level operand value	Total capacity of performance trace information files	
	Number of SQL statements executed per second is about 200	Number of SQL statements executed per second is about 500
`00000007`	750 MB (default value#)	750 MB (default value#)
`0000001f`	750 MB (default value#)	2 GB
`0000007f`	2 GB	5 GB

  #: This is *default value of the pd_prf_file_count operand* × *default value of the pd_prf_file_size operand.*

- To approximate the total capacity of the performance trace information files required per unit, use the following guidelines:

  Total capacity of the performance trace information files = MAX(768,000, $a \times b \times c \times d$) (kilobytes)

  a: Number of front-end and back-end servers per unit

  For a HiRDB/Single Server or if there is only a system manager or a dictionary server, the value is 1.

  b: Number of SQL statements executed per second on each server

  You can obtain the average value per editing time by dividing `TOTAL`, the total SQL statement execution count in the SQL-related statistical information, by the editing time `EDIT TIME`.

  c: Average amount of performance trace collected per SQL statement (kilobytes)

  Assume one of the following values, based on the value of the `pd_prf_level` operand:

  ● `00000007`: 0.5 kilobytes

  ● `0000001f`: 0.5 kilobytes

  ● `0000007f`: 1 kilobyte

  Note that the above value varies according to conditions, such as the type of SQL statement, amount of data, system configuration, whether facilities are applied, and tuning status.

  d: Length of time for which the performance trace information is to be retained (seconds)

- The length of time for which the performance trace information can be retained per file might change drastically depending on numerous factors, such as the type of transaction and HiRDB configuration. To obtain accurate values, measure the time for which performance trace information is retained per file when the most typical transactions and utilities are executed. You can determine the time value by using the timestamp or the `pdprfed` command's editing results. Calculate the required number of file generations based on the measured time.

Notes

- This operand value affects the amount of disk space required under the HiRDB directory. The following amount of disk space is required to store the performance trace information files:

  pd_prf_file_count operand value × *pd_prf_file_size operand value*
- This operand affects the capacity of the error information files that are collected by using commands such as `pdinfoget`.
- If you change the value of this operand to a smaller value, the performance trace information files corresponding to the eliminated portion of the value will no longer be used, but they are not deleted automatically. If you need to delete any unneeded performance trace information file, delete them after HiRDB has been terminated.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `15` is assumed.

Relationship to other operands

This operand is related to the `pd_prf_file_size` operand.

35) pd_prf_file_size = *size-of-a-performance-trace-information-file*

~<unsigned integer> ((1024-1048576))(kilobytes)

Specifies the size of each performance trace information file.

Specification guidelines

Normally, this operand does not need to be specified. Consider specifying this operand if you want to retain the performance trace information files for a longer than normal time.

Notes

- This operand value affects the amount of disk space required under the HiRDB directory. The following amount of disk space is required to store the performance trace information files:

  pd_prf_file_count operand value × *pd_prf_file_size operand value*
- This operand affects the capacity of the error information files that are collected by using commands such as `pdinfoget`.
- If you increase this operand value, the time required by the `pdprfed` command to edit and output performance traces increases.
- If this operand value is too large, sending the files to the support service for purposes such as an investigation of performance issues might be difficult, even if the files are compressed. Specify the appropriate file size taking into account file transmission.
- The performance trace information is output to the performance trace information file every 10 seconds. If many traces are output at one time, the performance trace information file might become smaller in size by a maximum of 10 megabytes.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `51200` is assumed.

Relationship to other operands

This operand is related to the `pd_prf_file_count` operand.

3.2.15 Operands related to troubleshooting information

36) pd_cancel_dump = put | noput

This operand is designed to reduce the amount of troubleshooting information to be output.

Specifies whether troubleshooting information is to be collected in the following cases:

- When SQL does not terminate within the monitoring time specified by the `PDCWAITTIME` operand in the client environment definition
- When a UAP being executed is canceled by the `pdcancel` command

For details about the troubleshooting information to be collected, see the *HiRDB Version 9 System Operation Guide* and *Table 3-2 Error information that is displayed in the event of abnormal termination* in the section on the `pd_dump_suppress_watch_time` operand.

`put`:

Troubleshooting information is collected. Because the troubleshooting information is output to files under `%PDDIR%\spool`, a space shortage might occur in the file system.

Note that the troubleshooting information that has been collected is automatically deleted by HiRDB at the following timings.

- Every 24 hours while HiRDB is running (the deletion interval can be changed using the `pd_spool_cleanup_interval` operand).
- When HiRDB is started (whether to delete the troubleshooting information can be changed using the `pd_spool_cleanup` operand).

For the HiRDB administrator to delete troubleshooting information, the administrator must execute the `pdcspool` command.

`noput`:

Do not collect troubleshooting information. Because no troubleshooting information will be collected, the load on the file system is reduced. Specify this option if UAP cancellation occurs frequently during normal operation and there is no need to investigate the causes.

Troubleshooting information might be acquired even though noput is specified in this operand. For details, see *Table 3-2 Error information that is displayed in the event of abnormal termination* in the section on the `pd_dump_suppress_watch_time` operand.

For details about error information that is output in the event of abnormal termination, see *Table 3-2 Error information that is displayed in the event of abnormal termination* in the section on the `pd_dump_suppress_watch_time` operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `put`.

37) pd_dump_suppress_watch_time = *troubleshooting-information-output-suppression-time*

~<unsigned integer>((0-3600)) (seconds)

This operand is designed to reduce the amount of troubleshooting information to be output.

This operand specifies the amount of time (in seconds) during which to suppress outputting again the troubleshooting information (files under `%PDDIR%\spool`) that is output when any of the following situations occurs.

- The time specified in `PDCWAITTIME` is exceeded.
- The UAP being executed is cancelled by the `pdcancel` command.
- A process terminates abnormally.

Once troubleshooting information is output, no troubleshooting information is output again until the time specified by this operand has elapsed. For example, if `60` is specified for this operand, no troubleshooting information is output again until 60 seconds have passed because troubleshooting information was previously output.

Note that if `0` is specified for this operand, outputting of troubleshooting information is not suppressed.

Advantage

When there are multiple HiRDB server processes, timing out, for example, might cause them to terminate abnormally one after the other. If abnormal terminations of server processes occur successively, troubleshooting information, such as core files and simple dumps, are repeatedly collected, thus causing a space shortage on the disk on which the HiRDB directory is located. If such a shortage occurs, HiRDB might terminate abnormally. Therefore, specify this operand to make sure that no disk space shortage occurs.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

Notes

- When abnormal termination is caused by an internal conflict, or if a signal is received from outside, troubleshooting information is collected regardless of the value specified for this operand.
- Table 3-1 lists the error information that is displayed in the event of abnormal termination.

Table 3–2: *Error information that is displayed in the event of abnormal termination*

Cause of abnormal termination		Error information	pd_dump_suppress_watch_time value			
			0		Not 0	
			pd_cancel_dump value			
			put	noput	put	noput
`PDSWAITTIME` exceeded		Save core file	N	N	N	N
		Snapshot of error	N	N	N	N
		Simple dump file	N	N	N	N
		`KFPA20009-W` message	N	N	N	N
		SQL runtime warning information file	N	N	N	N
`PDSWATCHTIME` exceeded		Save core file	N	N	N	N
		Snapshot of error	N	N	N	N
		Simple dump file	N	N	N	N
		`KFPA20009-W` message	N	N	N	N
		SQL runtime warning information file	N	N	N	N
`PDCWAITTIME` exceeded	`pd_client_waittime _over_abort = Y`	Save core file	Y	Y	Y+	Y+
		Snapshot of error	Y	Y	Y+	Y+
		Snapshot 2 of error	Y	Y	Y+	Y+
		Simple dump file	Y	Y	Y+	Y+
		`KFPA20009-W` message	Y	Y	Y+	Y+
		SQL runtime warning information file	Y	Y	Y+	Y+
		Shared memory dump file	F	F	F	F
	`pd_client_waittime _over_abort = N`	Save core file	N	N	N	N
		Snapshot of error	Y	N	Y+	N
		Snapshot 2 of error	N	N	N	N
		Simple dump file	Y	N	Y+	N
		`KFPA20009-W` message	Y	N	Y+	N
		SQL runtime warning information file	Y	N	Y+	N
		Shared memory dump file	N	N	N	N

Cause of abnormal termination	Error information	pd_dump_suppress_watch_time value			
		0		Not 0	
		pd_cancel_dump value			
		put	noput	put	noput
`pdcancel` command	Save core file	N	N	N	N
	Snapshot of error	Y	N	Y+	N
	Simple dump file	Y	N	Y+	N
	`KFPA20009-W` message	Y	N	Y+	N
	SQL runtime warning information file	Y	N	Y+	N
Internal `kill9`#1	Save core file	N	N	N	N
	Snapshot of error	Y	N	Y+	N
	Simple dump file	Y	N	Y+	N
	`KFPA20009-W` message	Y	N	Y+	N
	SQL runtime warning information file	Y	N	Y+	N
Internal `kill3`#2	Save core file	Y	Y	Y+	Y+
	Snapshot of error	Y	Y	Y+	Y+
	Simple dump file	Y	Y	Y+	Y+
	`KFPA20009-W` message	Y	Y	Y+	Y+
	SQL runtime warning information file	Y	Y	Y+	Y+
Abort#3	Save core file	Y	Y	Y+	Y+
	Snapshot of error	Y	Y	Y+	Y+
	Simple dump file	Y	Y	Y+	Y+
	`KFPA20009-W` message	Y	Y	Y+	Y+
	SQL runtime warning information file	Y	Y	Y+	Y+
	Abort information file	Y	Y	Y+	Y+
Other#4	Save core file	D	D	D	D
	Snapshot of error	Y	Y	Y	Y
	Simple dump file	D	D	D	D
	`KFPA20009-W` message	D	D	D	D
	SQL runtime warning information file	D	D	D	D

Legend:

Y: Outputs error information. The specification of the `pd_dump_suppress_watch_time` operand is invalid.

N: Does not output error information.

Y+: Outputs error information. The specification of the `pd_dump_suppress_watch_time` operand is valid.

D: Error information might not be output depending on how the process is terminated.

F: After the unit is started, error information is output during the first dump. The units that output shared memory dumps can be restricted by specifying `shm_fesonly` in the `pd_clt_waittime_over_dump_level` operand.

#1

This refers to `SIGKILL` being issued internally, as occurs when a UAP is abnormally terminated by OpenTP1. It does not include abnormal termination due to exceeding the `PDCWAITTIME` value or due to issuance of the `pdcancel` command.

#2

This refers to `SIGQUIT` being issued internally, as occurs when an abnormality is detected. It does not include abnormal termination due to exceeding the `PDCWAITTIME` value or due to issuance of the `pdcancel` command.

#3

This refers to HiRDB detecting an inconsistency and calling `abort()`.

#4

This refers to an unforeseen error, such as `SIGSEGV`, `SIGBUS`, receiving an external signal, or `exit`.

38) pd_spool_cleanup_interval = *troubleshooting-information-deletion-interval*

~<unsigned integer>((0-744)) (times)

This operand is used for deleting the troubleshooting information and temporary work files that have been output. If these items are not deleted, they might cause a space shortage on the disk on which the HiRDB directory is located. If such a shortage occurs, HiRDB might terminate abnormally. Therefore, HiRDB regularly deletes the following files:

- Troubleshooting information file (files in `%PDDIR%\spool`)
- Temporary work files (files in `%PDDIR%\tmp`)

This operand specifies the deletion interval (hours). For example, if `48` is specified for this operand, these files are deleted every 48 hours. Normally, (if this operand is omitted) files are deleted every 24 hours.

Note that time counting begins when HiRDB is normally started. When HiRDB is normally terminated, time counting also stops. Then, the count returns to 0 during the next normal startup.

Specify the files to be deleted using the `pd_spool_cleanup_interval_level` operand explained as follows.

Operand rule

If `0` is specified, files are not deleted.

Specification guidelines

If `24`, `48`, `72`, and so on are specified for this operand, files are deleted at the predetermined time. Specify the time so that files are deleted during the time period that does not overload the system.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `24`.

Notes

Even while HiRDB is stopped because of planned termination, forced termination, or abnormal termination, time counting continues. However, if the deletion time arrives while HiRDB is stopped, files are not deleted. Files are not deleted until the next deletion time. To restart HiRDB after deleting the files, execute the `pdcspool` command.

Remarks

The difference between the `pd_spool_cleanup_interval` and `pd_spool_cleanup` operands is as follows:

- The `pd_spool_cleanup_interval` operand is related to regular deletion of troubleshooting information.
- The `pd_spool_cleanup` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup` operand.

39) pd_spool_cleanup_interval_level = *number-of-days* [, *deletion-type*]

This operand is used for deleting the troubleshooting information and temporary work files that have been output, and specifies the condition for regularly deleting the troubleshooting information and temporary work files.

***number-of-days:* ~<unsigned integer>((1-24855)) (days)**

Troubleshooting information files that are older than the number of days specified here are deleted. For example, if 3 is specified, all troubleshooting information files, except for those created within the last 3 days (or 3 days × 24 hours = 72 hours), are deleted.

***deletion-type:* <character string>**

Specifies the type of troubleshooting information file to be deleted.

all: All files are to be deleted.

dump: Only the files internally acquired by HiRDB are to be deleted.

The following are the types of troubleshooting information files that are deleted.

Troubleshooting information file type	Directory name	all	dump	Remarks
Deadlock and timeout information	pdlckinf	Y	N	Output when an error occurs during locking.
Access path information	pdsqldump	Y	N	Output when the access path display utility is used.
Save core file	save	Y	Y	Output when a process is abnormally terminated.
Shared memory dump file	pdshmdump	Y	Y	Output when a process or unit is abnormally terminated.
Simple dump file	pdsysdump	Y	Y	None
	pdsdsdump	Y	Y	Nonexistent in a HiRDB/Parallel Server
	pdfesdump pddicdump pdbesdump	Y	Y	Nonexistent in a HiRDB/Single Server
System log file status information file	pdjnlinf	Y	N	Files under \pdjnlinf\errinf are not deleted.

Y: File is deleted.

N: File is not deleted.

Note: Directory names under %PDDIR%\spool are shown.

All temporary work files, except for those listed as follows, are deleted regardless of the deletion type specification. Parentheses indicate directory names under %PDDIR%\tmp.

- Current working directory (home) of the process in which HiRDB is to start
- Shared memory information file (pdommenv)
- Differential information files of the pdbufls command (files with names that begin with CMb)

Condition

A value other than 0 must be specified for the pd_spool_cleanup_interval operand.

Specification guidelines

- Specify a value that is longer than the execution time of commands (including utilities). For example, if the execution of the pdcopy command, which collects backup data, requires 24 hours (1 day), specify at least 2 for the number of days. If you do not specify a value that is longer than the execution time of the command, the temporary work files being used by the command are deleted, and thus the command might not run correctly.
- If the specified value is too large, disk space might fill up; if the specified value is too small, information files needed for troubleshooting might be deleted.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the following values are assumed:

- *number-of-days*: `7`
- *deletion-type*: `all` (or `dump` when `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand)

Operand rule

If you specify a deletion type, you must also specify a number of days.

Note

If the `TMP` environment variable is specified but the `pd_tmp_directory` operand is not specified, temporary work files used by commands and utilities will be output to the directory specified in the `TMP` environment variable. The temporary work files that are output to the directory specified by the `TMP` environment variable are not subject to regular deletion. Therefore, use Windows Explorer, for example, to delete them.

Remarks

The difference between the `pd_spool_cleanup_interval_level` and `pd_spool_cleanup_level` operands is as follows:

- The `pd_spool_cleanup_interval_level` operand is related to regular deletion of troubleshooting information.
- The `pd_spool_cleanup_level` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval_level` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup_level` operand.

40) pd_spool_cleanup = normal | force | no

This operand is used for deleting the troubleshooting information that has been output.

Specifies whether troubleshooting information files (files under `%PDDIR%\spool`) that were output previously by HiRDB are to be deleted when HiRDB is started. This operand is related to the `pd_spool_cleanup_level` operand, described as follows.

`normal`:

Delete the files when HiRDB is started normally or is restarted following a planned termination.

`force`:

Delete the files whenever HiRDB is started, regardless of the HiRDB activation mode.

`no`:

Do not delete the files.

Specification guidelines

If troubleshooting information files take up too much disk space, specify `normal` or `force`.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `force`.

Remarks

The difference between the `pd_spool_cleanup_interval` and `pd_spool_cleanup` operands is as follows:

- The `pd_spool_cleanup_interval` operand is related to regular deletion of troubleshooting information.
- The `pd_spool_cleanup` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup` operand.

41) pd_spool_cleanup_level = *number-of-days* [, *deletion-type*]

This operand is used for deleting the troubleshooting information that has been output, and specifies the condition for deleting the troubleshooting information files during HiRDB startup.

number-of-days: **~<unsigned integer>((0-24855)) (days)**

Specifies a number of days when troubleshooting information that is older than the specified number of days is to be deleted. For example, if `3` is specified, all troubleshooting information will be deleted except for the information that is fewer than 3 days old (3 days × 24 hours = 72 hours).

If `0` is specified, all troubleshooting information files are deleted.

deletion-type: **<character string>**

Specifies the type of troubleshooting information to be deleted.

`all`: Delete all file types.

`dump`: Delete only files collected internally by HiRDB.

The following are the types of troubleshooting information files that are deleted:

Troubleshooting information file type	Directory name	all	dump	Remarks
Deadlock and timeout information	`pdlckinf`	Y	N	Output when an error occurs during locking.
Access path information	`pdsqldump`	Y	N	Output when the access path display utility is used.
Save core file	`save`	Y	Y	Output when a process is abnormally terminated.
Shared memory dump file	`pdshmdump`	Y	Y	Output when a process or unit is abnormally terminated.
Simple dump file	`pdsysdump`	Y	Y	None
	`pdsdsdump`	Y	Y	Nonexistent in a HiRDB/Parallel Server

	`pdfesdump`
	`pddicdump`
	`pdbesdump`
		Y	Y	Nonexistent in a HiRDB/Single Server

System log file status information file	`pdjnlinf`	Y	N	Files under `\pdjnlinf\errinf` are not deleted.

Y: File is deleted.

N: File is not deleted.

Note: Directory names under `%PDDIR%\spool` are shown.

Condition

`normal` or `force` (default value) must be specified for the `pd_spool_cleanup` operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the following values are assumed:

- *number-of-days*: `7`
- *deletion-type*: `all` (or `dump` when `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand)

Operand rule

A number of days and a deletion type must both be specified.

Remarks

The difference between the `pd_spool_cleanup_interval_level` and `pd_spool_cleanup_level` operands is as follows:

- The `pd_spool_cleanup_interval_level` operand is related to regular deletion of troubleshooting information.
- The `pd_spool_cleanup_level` operand is related to the deletion of troubleshooting information during HiRDB startup.

Therefore, if you plan to run HiRDB continuously for 24 hours, consider specifying the `pd_spool_cleanup_interval_level` operand. If you plan to terminate HiRDB every day, consider specifying the `pd_spool_cleanup_level` operand.

42) pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*

~<unsigned integer>((126-16383))

A HiRDB process records the history of the executed functions and macros inside the process private memory. This history is called a *module trace*. This operand specifies the number of module trace records. The content of this history is loaded into the `core` file and is output when a process error occurs.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `126`.

Note

Process private memory of the following size is allocated to each process:

In the 32-bit mode: 64 + 48 × `pd_module_trace_max` *operand value* (bytes)

In the 64-bit mode: 64 + 64 × `pd_module_trace_max` *operand value* (bytes)

43) pd_module_trace_timer_level = 0 | 10 | 20

Specifies how to acquire the time to be output in module traces. The following table explains the meaning of the value specified for this operand.

Specified value	Time acquisition method
`0`	Time is output in seconds at every module trace output location.
`10`	Time is output in microseconds only at performance-critical module trace output locations, such as those before and after input/output processing, and time is output in seconds at other locations.
`20`	Time is output in microseconds at every module trace output location.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

Note

If you specify a value other than `0` for this operand, a function for acquiring time in microseconds is issued, and as a result, system performance might decline.

44) pd_pth_trace_max = *maximum-number-of-stored-communication-traces*

~<unsigned integer> ((1024-8388608))

Specifies the maximum number of communication trace records to be used as troubleshooting information.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reason such as performance checking, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `1024`.

Notes

Increasing the value of this operand increases the process private memory secured by HiRDB processes.

Process private memory for communication traces is calculated based on this operand's value rounded up to the power of two. For details about memory requirements, see *Calculation of required memory* in the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_pth_trace_max` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

3.2.16 Operands related to global buffers

45) `pd_dbbuff_wait_interval` = ***global-buffer-occupation-state-check-interval***

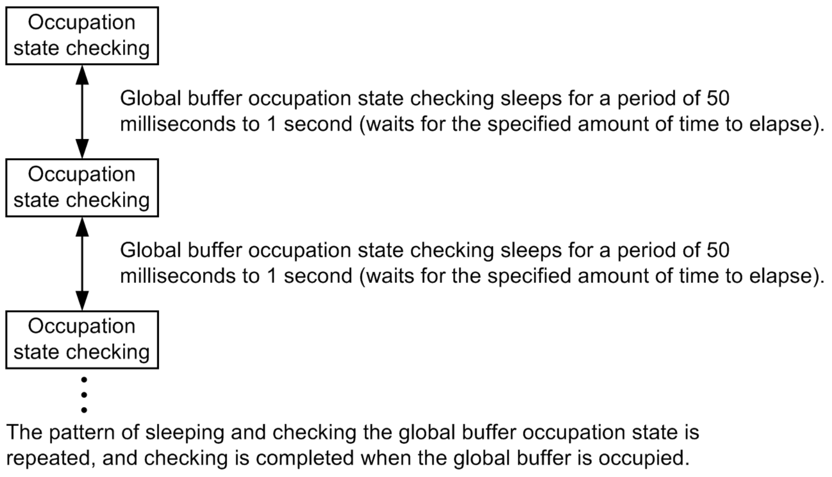
~<unsigned integer>((0-2147483647))(milliseconds)

This operand is applicable only to HiRDB/Parallel Server.

Specifies the interval at which the global buffer occupation state is to be checked. Specifying this operand changes the method of checking the global buffer occupation state.

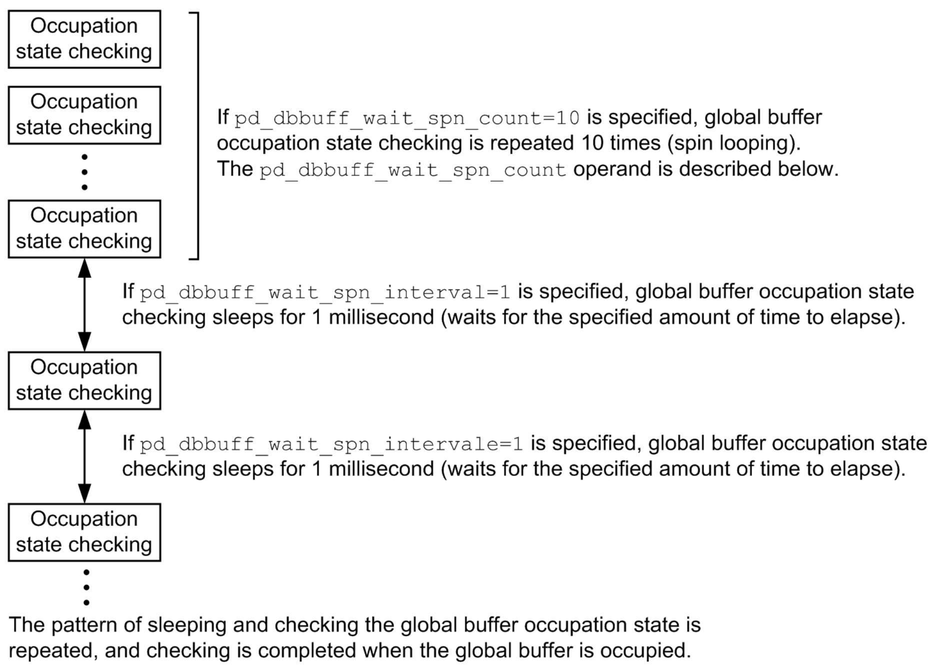
When this operand is not specified

The following processing occurs:


Occupation state checking
Global buffer occupation state checking sleeps for a period of 50 milliseconds to 1 second (waits for the specified amount of time to elapse).
Occupation state checking
Global buffer occupation state checking sleeps for a period of 50 milliseconds to 1 second (waits for the specified amount of time to elapse).
Occupation state checking
The pattern of sleeping and checking the global buffer occupation state is repeated, and checking is completed when the global buffer is occupied.

When this operand is specified

The following processing occurs:


Occupation state checking
Occupation state checking
Occupation state checking
If pd_dbbuff_wait_spn_count=10 is specified, global buffer occupation state checking is repeated 10 times (spin looping).
The pd_dbbuff_wait_spn_count operand is described below.
If pd_dbbuff_wait_spn_interval=1 is specified, global buffer occupation state checking sleeps for 1 millisecond (waits for the specified amount of time to elapse).
Occupation state checking
If pd_dbbuff_wait_spn_intervale=1 is specified, global buffer occupation state checking sleeps for 1 millisecond (waits for the specified amount of time to elapse).
Occupation state checking
The pattern of sleeping and checking the global buffer occupation state is repeated, and checking is completed when the global buffer is occupied.

Specification guidelines

Specify this operand when all of the conditions listed below are satisfied. Performance might improve. Typically when this operand is used, a value of 1 is specified.

- Global buffer lock-release wait has occurred. (You can check for this based on `WAITL` in the execution result of the `pdbufls` command.)
- You want to improve performance, even if doing so increases the CPU usage rate.

If the CPU usage has become too high because 1 was specified in this operand, increase the value. If there is unused capacity in the CPU usage rate when 1 is specified in this operand, increase the `pd_dbbuff_wait_spn_count` operand value. Performance might improve.

46) pd_dbbuff_wait_spn_count = *maximum-spin-loop-count-for-global-buffer-occupation-state-checking*

~<unsigned integer>((0-2147483646))

This operand is applicable only to HiRDB/Parallel Server.

Specifies the maximum spin loop count in an interval loop that can occur during global buffer occupation state checking. For details, see the description of the `pd_dbbuff_wait_interval` operand.

Specification guidelines

Normally, there is no need to specify this operand. Specify this operand when you specify 1 in the `pd_dbbuff_wait_interval` operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

3.2.17 Operands related to unit status files

47) pd_syssts_file_name_1 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

:

pd_syssts_file_name_7 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

These operands define status files for the unit.

Although the `pd_syssts_file_name_2` to `7` operands can be omitted, the `pd_syssts_file_name_1` operand cannot be omitted.

"*logical-file-name*" ~<identifier>((1-8 characters))

Specifies the logical file name of a status file for the unit.

When a command that manipulates the status file is to be executed, the logical file name defined here must be specified.

"*file-a-status-file-name*" ~<path name>((maximum of 167 characters))

Specifies the name of the File A status file as an absolute path name.

"*file-b-status-file-name*" ~<path name>((maximum of 167 characters))

Specifies the name of the File B status file as an absolute path name.

Specification guidelines

- The files specified as File A and File B must be status files created with the `pdstsinit` command. If a file that has not been created with the `pdstsinit` command is specified, a virtual status file is created.
- If an error occurs in a status file, HiRDB swaps the status files. If no spare file is available for swapping, HiRDB (or a unit for a HiRDB/Parallel Server) terminates abnormally. For this reason, defining a large number of status files improves system reliability, but at the expense of increasing the amount of disk space that is required.
- The status files specified for File A and File B must have the same record length and capacity.

Operand rules

- Up to seven instances of this operand can be specified.
- A status file has a dual structure consisting of File A and File B; both must be specified.
- Environment variables cannot be used for the absolute path names of the File A and File B file names.
- The same names cannot be specified for the logical file name, the Status File A file name, and the Status File B file name.
- HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\usts01`, `C:\hirdb\sysfile` is not case sensitive, but `usts01` is case sensitive.

Notes

- When HiRDB is started normally, the primary file (the file that was the primary file when HiRDB was terminated) is inherited. However, if there is no current file that can be inherited, for example because all status files have been initialized, the first status file that was specified among those specified by the `pd_syssts_file_name_1` to `7` operands becomes the current file. The remaining files become spare files if they can be opened. Any files that cannot be opened become reserved files.
- When HiRDB is restarted, the primary file (the file that was the primary file when HiRDB was terminated) is inherited.

Using a virtual status file

Specifying a virtual status file makes it possible to add a new status file while HiRDB is running.

For example, if the number of spare files is reduced because of errors in status files, virtual status files can be converted into spare files. The procedure for converting virtual status files into spare files follows.

Procedure

1. Use the `pdstsinit` command to create a status file in a HiRDB file system area for system files.
2. Use the `pdstsopen` command to open the status file.

These operations can be performed while HiRDB is running; there is no need to stop HiRDB.

Advantages and disadvantages

Defining a virtual status file reduces the amount of space required in the HiRDB file system area. However, a virtual status file cannot be added as a spare file if the HiRDB file system area for system files does not have sufficient space (sufficient space for adding the file), thus reducing system reliability.

When virtual status files are not defined, the amount of space required in the HiRDB file system area is increased. However, because a swap destination is guaranteed when a file error occurs, system reliability is increased.

Notes

When virtual status files are defined, HiRDB determines that a status file error has occurred when the unit is started. For this reason, HiRDB cannot start if `stop` (default value) is specified for the `pd_syssts_initial_error` operand. When virtual status files are defined, specify `continue` or `excontinue` for the `pd_syssts_initial_error` operand. It is also necessary before starting HiRDB to specify the current file in the `pd_syssts_last_active_file` operand.

3.2.18 Operands related to unit status files (when an error occurs)

For details about the measures to be taken when an error occurs in a status file, see the *HiRDB Version 9 System Operation Guide*.

48) pd_syssts_initial_error = <u>stop</u> | continue | excontinue

When HiRDB (or a unit) starts, HiRDB performs a process of identifying the current unit status file. This operand specifies the action that HiRDB takes when any of the following errors is detected during this identification process.

- No real unit status file is found.
- An error is detected in the unit status file.

Note that the current file identification process is applied to the unit status files specified by the `pd_syssts_file_name_1` to `7` operands.

`stop`:

When an error is detected in the unit status file during the current file identification process, startup of HiRDB (or the unit) is stopped. In this case, first take a corrective action for the status file in which the error was detected, and then start HiRDB.

`continue` or `excontinue`:

Even when an error is detected in the unit status file during the current file identification process, startup of HiRDB (or the unit) is continued if the current file is normal. However, startup might be stopped depending on the value specified for the `pd_syssts_singleoperation` operand (whether operation continues with a single status file). The following table shows the relationship to the `pd_syssts_singleoperation` operand.

● **Relationship to the pd_syssts_singleoperation operand**

pd_syssts_singleoperation operand value	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
`continue`	When an error is detected in the unit status file, HiRDB cannot identify the current file, and thus the startup of HiRDB (or the unit) is stopped.	The HiRDB administrator identifies the current file and specifies the `pd_syssts_last_active_file` and `pd_syssts_last_active_side` operands. Afterwards, start HiRDB (or the unit).
`stop` (default value)	When an error is detected in the unit status file, HiRDB identifies the current file, and the startup of HiRDB (or the unit) is continued. However, if File A or B satisfies any of the conditions listed in the following table (cases in which HiRDB cannot identify the current file), the startup of HiRDB is stopped.	If HiRDB cannot identify the current file, the HiRDB administrator identifies the current file and specifies the `pd_syssts_last_active_file` and `pd_syssts_last_active_side` operands. Afterwards, start HiRDB (or the unit).

● **When HiRDB cannot identify the current file**

pd_syssts_initial_error operand value	File A status	File B status
`continue`	Error shutdown	Error shutdown
	Error shutdown	Open (initial state)
	Error shutdown	No real file
	Open (initial state)	Error shutdown

pd_syssts_initial_error operand value	File A status	File B status
	Open (initial state)	No real file
	No real file	Open (initial state)
	No real file	Error shutdown
	No real file	No real file
`excontinue`	Error shutdown	Error shutdown
	Error shutdown	No real file
	No real file	Error shutdown
	No real file	No real file

Therefore, to minimize actions by the HiRDB administrator (to reduce the number of cases in which both the `pd_syssts_last_active_file` and `pd_syssts_last_active_side` operands must be specified), specify the following:

- `pd_syssts_initial_error = excontinue`
- `pd_syssts_singleoperation = stop`

Specification guidelines

The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_syssts_initial_error operand value	
	stop	continue or excontinue
Processing by HiRDB during HiRDB or unit startup	When an error is detected in a unit status file, the startup of HiRDB (or the unit) is stopped.	Even when an error is detected in a unit status file, the startup of HiRDB (or the unit) is continued if the current file is normal.
Specification guideline	To improve system reliability, specify `stop`.	To simplify the error-handling actions during HiRDB startup, specify `continue` or `excontinue`.
Advantage	Guarantees that all unit status files are normal when HiRDB starts. Therefore, if an error occurs in the current file after HiRDB has started, it can be swapped to a spare file.	Even when an error is detected in a unit status file during HiRDB startup, HiRDB can start with the remaining normal files only. Therefore, HiRDB stop time can be shortened. In this case, because the number of spare files has become small, it is necessary to immediately repair the status files containing errors.
Disadvantage	Possibility increases that an error in a unit status file stops the startup of HiRDB.	Because HiRDB might be running with only a small number of spare files, system reliability is low. Depending on the number of spare files available, it might not be possible to swap unit status files.

Notes

- If both current files are abnormal, the startup of HiRDB (or the unit) is stopped regardless of the value specified for this operand.
- Before starting HiRDB, do not initialize the current file with the `pdstsinit` command. If the current file is initialized, HiRDB cannot be restarted.
- If `excontinue` is specified for this operand and all generations of the status files are in single-operation mode and open (initial status), HiRDB does not start.

Remarks

The following figure shows the values specified for this operand, the processing performed by HiRDB, and the actions to be taken by the HiRDB administrator.

- Values specified for this operand and the processing performed by HiRDB

pd_syssts_initial_error operand value	Status file error	pd_syssts_singleoperation operand value	Can HiRDB identify the current file?	Specification of pd_syssts_last_active_file	Does the pd_syssts_last_active_file operand value match the latest file that can be opened?	Current file error	Specification of the pd_syssts_last_active_side operand	Is the file specified for the pd_syssts_last_active_side operand usable?	Numbers listed in the Table Processing performed by HiRDB and actions to be taken by the HiRDB administrator
`stop` (default value)	None	—	—	—	—	—	—	—	[1]
	Yes	—	—	—	—	—	—	—	[5]
`continue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
`excontinue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]

Legend:

—: Not applicable (the condition does not affect the processing performed by HiRDB)

- Processing performed by HiRDB and actions to be taken by the HiRDB administrator

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[1]	HiRDB startup processing continues.	None
[2]	HiRDB identifies the latest current file and continues the startup processing.	Make the file that is in the error-shutdown state into a spare file.
[3]	Using the file specified in the `pd_syssts_last_active_file` operand as the current status file, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[4]	Using the files specified in the `pd_syssts_last_active_file` and `pd_syssts_last_active_side` operands as the current status files, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[5]	Because `stop` is specified for the `pd_syssts_initial_error` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000010` in message `KFPS01005-E`.

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[6]	Because the current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000016` in message `KFPS01005-E`.
[7]	Because the normal current file identified by HiRDB does not match the file specified in the `pd_syssts_last_active_side` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000017` in message `KFPS01005-E`.
[8]	Because the current file name identified by HiRDB does not match the file name specified in the `pd_syssts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000015` in message `KFPS01005-E`.
[9]	Because the normal current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000018` in message `KFPS01005-E`.

49) pd_syssts_singleoperation = stop | continue

Specifies whether processing of unit status files continues in the single-operation mode.

The status file single-operation mode means that processing is to continue using only the normal file (single file) when an error occurs in the status file and a spare file is not available. For details about the status file single-operation mode, see the *HiRDB Version 9 Installation and Design Guide*.

If an error occurs in one of the current files and a spare file is available, the status file is swapped and processing continues regardless of the value specified for this operand (operation in the single-operation mode does not occur).

`stop`:

Do not permit operation in the single-operation mode. If operation in the single-operation mode is necessary, HiRDB (or a unit for a HiRDB/Parallel Server) terminates abnormally. If HiRDB terminates abnormally, allocate spare files and then restart HiRDB.

`continue`:

Enable operation in the single-operation mode. When the single-operation mode goes into effect, the message `KFPS01044-I` is output. If an error occurs in the normal file during operation in the single-operation mode, or if HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated while the status file is being updated, HiRDB cannot be restarted. Therefore, when the single-operation mode goes into effect, immediately allocate spare files.

Specification guidelines

- We recommend that you specify `stop` to increase system reliability. We also recommend that you increase the number of spare files to guard against errors in the current files.
- The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_syssts_singleoperation operand value	
	stop	continue
Specification guideline	To improve system reliability, specify `stop`.	Specify `continue` if it is important not to stop HiRDB.
Advantage	When an error occurs in one of the current files and a spare file is not available, operation in the single-operation mode does not occur, and HiRDB is abnormally terminated. Consequently, the possibility of losing the content of the current files is reduced.	Even when an error occurs in one of the current files and a spare file is not available, processing can be continued. Therefore, the possibility that an error in the status file stops HiRDB is reduced.
Disadvantage	Possibility that an error in the status file stops HiRDB increases. However, increasing the number of spare files can reduce this possibility.	If an error occurs in the normal status file during operation in the single-operation mode, or if HiRDB is abnormally terminated during updating of the

Item	pd_syssts_singleoperation operand value	
	stop	continue
		status file, the content of the current file is lost and HiRDB cannot be restarted.

Relationship to other operands

The combination of the values specified for the `pd_syssts_singleoperation` and `pd_syssts_initial_error` operands determines the processing by HiRDB when an error occurs in the status file. Therefore, the values to be specified for these operands must be determined together.

50) pd_syssts_last_active_file = "*logical-file-name*"

~<identifier>((1-8 characters))

Specifies the name of the logical file name of the status file to be used as the current file at the time of HiRDB (or a unit for a HiRDB/Parallel Server) startup.

HiRDB compares the file specified in this operand with the file selected by HiRDB to be the current file. If they match, HiRDB is started; otherwise, HiRDB is not started.

Conditions

The following conditions must be satisfied:

- `continue` or `excontinue` is specified for the `pd_syssts_initial_error` operand.
- It cannot be determined if the current file selected by the HiRDB system was the most recent current file during the previous session.

Specification guidelines

1. To start HiRDB immediately after initializing all status files:
   From among the operable logical files specified in the `pd_syssts_file_name_1-7` operands, specify the one that has the smallest number. Forced startup will be used in this case, regardless of the previous termination mode.
2. When both of the current status files are normal:
   Specify the name of the current file#.
   If HiRDB cannot be started even though the name of the current file is specified, the current file might have been initialized. In this case, first initialize all status files, and then use the method in step 1 above to start HiRDB (forced startup will be used, regardless of the previous termination mode).
3. When one of the current status files has an error:
   Use the method in step 2 above, with the following operands specified:
   - `pd_syssts_singleoperation = continue` or `excontinue`
   - `pd_syssts_last_active_side`
4. When both of the current status files have errors:
   Initialize all status files, then execute the method in step 1 above (forced startup will be used, regardless of the previous termination mode).
5. When a virtual status file is specified:
   Specify the name of the current file#.

#: The names of the current files (that were active at the end of the previous operation) can be determined from the following messages:

- `KFPS01001-I`
- `KFPS01010-E`
- `KFPS01011-I`
- `KFPS01063-I`

Of the status files displayed by these messages, the one that is reported in the message that was output most recently is the current file.

51) pd_syssts_last_active_side = A | B

Specify this operand if you want to start HiRDB (or a unit for a HiRDB/Parallel Server) when one of the current files is in an error state. Specify the normal status file for this operand. HiRDB compares the file specified in this operand with the file selected by HiRDB. If they match, HiRDB copies the contents of the normal status file to secondary File A and File B. Afterwards, the secondary file is switched to the current file and HiRDB is started. If the files do not match, HiRDB is not started.

Conditions

The following operands must be specified:

- `pd_syssts_initial_error = continue` or `excontinue`
- `pd_syssts_last_active_file`

3.2.19 Operands related to security

For details about how to use the security audit facility, see the *HiRDB Version 9 System Operation Guide*.

52) pd_audit = Y | N

Specifies whether to begin collecting an audit trail when HiRDB (or a unit for a HiRDB/Parallel Server) is started.

`Y`: Begins collecting an audit trail when HiRDB is started.

`N`: Does not begin collecting an audit trail when HiRDB is started.

Even if `N` is specified for this operand, you can still collect an audit trail by executing the `pdaudbegin` command.

Conditions

All of the following conditions must be satisfied. If `Y` is specified when all of these conditions are not satisfied, HiRDB (or a unit for a HiRDB/Parallel Server) cannot be started.

- A HiRDB file system area has been created for an audit trail file.
- The name of the HiRDB file system for the audit trail file is specified for the `pd_aud_file_name` operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `N`.

53) pd_aud_file_name = *HiRDB-file-system-area-name-for-audit-trail-file*

~<path name>((up to 150 characters))

This operand is required if you use the security audit facility. If you do not specify this operand, you cannot use the security audit facility.

Specify an absolute path name for the name of the HiRDB file system area for an audit trail file.

When you use the security audit facility on a HiRDB/Parallel Server, we recommend that you acquire an audit trail for the entire system. To do so, specify one of the following:

- `pd_aud_file_name` operand in the system common definition
- `pd_aud_file_name` operand in each unit control information definition

In system configurations that run multiple units on a single server machine, the `pd_aud_file_name` operand must be specified in all the unit control information definitions.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed.

Notes

- When this operand is specified, HiRDB (or a unit for a HiRDB/Parallel Server) cannot be started if an error occurs during the access to the HiRDB file system area for audit trail files.
- If the same audit trail file is specified in `pd_aud_file_name` operands in the system common definition for multiple units on the same server machine, the correct audit trail cannot be acquired.

54) pd_aud_max_generation_size = ***audit-trail-file-maximum-size***

~<unsigned integer>((1-5240)) (megabytes)

Specifies, in megabytes, the maximum size of audit trail files.

Specification guidelines

Because HiRDB needs 20 MB for management, determine the value for this operand so that the following condition is satisfied:

- *pd_aud_max_generation_size value* × *pd_aud_max_generation_num value* < *size of HiRDB file system area for audit trail files* (value of the `-n` option of the `pdfmkfs` command) - 20 MB
- When the specified value is smaller than the capacity of one audit trail record or no value is specified for this operand, and the size of one audit trail record is larger than the default, it will not be possible for HiRDB to start.

  To start such a HiRDB unit, set the operand value so that the following condition is satisfied:

  Value of pd_aud_max_generation_size ≥ ↑*Maximum audit trail record size* ÷ 1,024↑ × 1,024 + 2,048 (bytes)

  Use the following formula to calculate the audit trail record size:

  Maximum audit trail record size = 1,067 + ↑*value of pd_aud_sql_source_size* ÷ 4↑ × 4 + ↑*value of pd_aud_sql_data_size* ÷ 4↑ × 4 (bytes)
- If data is being output to the audit trail file asynchronously, see *Output to audit trail file (asynchronous output)* in the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `100`.

55) pd_aud_max_generation_num = ***maximum-audit-trail-file-count***

~<unsigned integer>((2-200))

Specifies the maximum number of (number of generations of) audit trail files to be created inside the HiRDB file system area for audit trail files.

Specification guidelines

- We recommend that you not specify the maximum value (`200`) in case errors occur in all audit trail files. For details about how to handle errors in audit trail files, see the *HiRDB Version 9 System Operation Guide*.
- Because HiRDB needs 20 MB for management, determine the value for this operand so that the following condition is satisfied:

pd_aud_max_generation_size-value × *pd_aud_max_generation_num-value* < *size-of-HiRDB-file-system-area-for-audit-trail-files* (value of the `-n` option of the `pdfmkfs` command) - 20 MB

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `50`.

Notes

During the startup of HiRDB (or a unit for a HiRDB/Parallel Server), if there is a file with a generation number that is greater than the value specified for this operand, the specified value becomes invalid. In this case, the largest generation number is assumed as the maximum number of audit trail files to be created inside the HiRDB file system area.

56) pd_aud_async_buff_size = ***size-of-buffer-used-for-asynchronous-output-of-audit-trail-file***

~<unsigned integer>((0, 4096-6553600)) (bytes)

Specifies the size (in bytes) of the buffer to be used for asynchronously outputting audit trail. If `0` is specified, audit trail is synchronously output. When the specified value is smaller than the maximum audit trail record size or no value is specified for this operand, and the maximum audit trail record size is larger than the default, it will not be possible for the HiRDB unit to start.

For details about the maximum audit trail record size, see the description of the `pd_aud_max_generation_size` operand.

The following table describes the advantages and disadvantages of each output method.

pd_aud_async_buff_size value	Audit trail output method	Advantages	Disadvantages
`0`	Synchronous output	Audit trail can be reliably output to an audit trail file.	Because file input/out occurs on the extension of SQL processing, the impact on performance is large.
`4096` to `6553600`	Asynchronous output	Can reduce the impact on SQL processing performance.	If HiRDB (or unit for a HiRDB/Parallel Server) is abnormally terminated after the audit trail is output to the buffer and before it is output to an audit trail file, the audit trail might be lost.

Specification guidelines

To output an audit trail asynchronously, we recommend that you set this buffer size on the large side. There is only one of these buffers per unit, so performance might be degraded if contention occurs among environments that have many transactions that create high processing loads.

Operand rule

For this operand, specify an integral multiple of 4,096. If a value that is not an integral multiple of 4,096 is specified, it is rounded up to an integer multiple of 4,096 and set as the value for this operand. For example, if `5000` is specified, `8192` is set for the operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `401408`. However, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `4096`.

Notes

- Starting HiRDB (or, for a HiRDB/Parallel Server, the unit) requires shared memory for the unit controllers equal in size to value of *pd_aud_async_buff_size* × *value of pd_aud_async_buff_count* (bytes). Make sure that the value from this equation does not exceed the upper limit for shared memory for unit controllers as a whole. For details about calculating the shared memory size used by unit controllers, see the *HiRDB Version 9 Installation and Design Guide*.
- When the values specified in the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands are small, all sectors of the asynchronous output buffer might wait for flushing, transaction execution times might lengthen, or, depending on the specification of the `pd_aud_no_standby_file_opr` operand, HiRDB (or the unit for a HiRDB/Parallel Server) might be forcibly terminated.

  Determine the settings for the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands taking into consideration the number of audit trail outputs per unit of time. For details, see *Output to audit trail file (asynchronous output)* in the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_aud_async_buff_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

57) pd_aud_async_buff_count = *number-of-buffer-sectors-used-for-asynchronous-output-of-audit-trail-file*

~<unsigned integer>((1-6500))

Specifies the number of buffer sectors to be used for asynchronously outputting an audit trail.

Specification guidelines

We recommend that the number of buffer sectors be set on the high side. If the value is too small, writing to buffers might take longer due to writing into audit trail files, which can degrade performance.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is the larger of the following two values:

- `1` (when there are no HiRDB servers in the unit)
- *Number of HiRDB servers in the unit* × 10

Notes

- Starting HiRDB (or, for a HiRDB/Parallel Server, the unit) requires shared memory for the unit controllers equal in size to value of *pd_aud_async_buff_size* × *value of pd_aud_async_buff_count* (bytes). Make sure that the value from this equation does not exceed the upper limit for shared memory for unit controllers as a whole. For details about calculating the shared memory size used by unit controllers, see the *HiRDB Version 9 Installation and Design Guide*.
- When the values specified in the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands are small, all sectors of the asynchronous output buffer might wait for flushing, transaction execution times might lengthen, or, depending on the specification of the `pd_aud_no_standby_file_opr` operand, HiRDB (or the unit for a HiRDB/Parallel Server) might be forcibly terminated.

  Determine the settings for the `pd_aud_async_buff_size` and `pd_aud_async_buff_count` operands taking into consideration the number of audit trail outputs per unit of time. For details, see *Output to audit trail file (asynchronous output)* in the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_aud_async_buff_count` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

58) pd_aud_async_buff_retry_intvl = *retry-interval-for-allocation-of-a-buffer-to-be-used-for-asynchronous-output-of-audit-trail-file*

~<unsigned integer>((1-1000)) (milliseconds)

Specifies the retry interval for monitoring for a buffer to be used for asynchronous output of the audit trail so that the audit trail can be acquired when all buffers are in use.

Specification guidelines

Normally, there is no need to specify this operand.

When the security audit facility is used and a UAP requires an extended amount of time to execute, specifying a small value in this operand might reduce the UAP execution time.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `50`.

59) pd_aud_sql_source_size = *size-of-sql-statement-output-to-audit-trail*

~<unsigned integer>((0-2000000))(bytes)

Specifies the size in bytes of the SQL statements output to the audit trail when using the security audit facility. When `0` is specified, no SQL statements are output to the audit trail. For SQL statements larger than the specified value, the portion in excess of the specified value is not output to the audit trail.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

Relationship to other operands

If you specify this operand, re-estimate the specifications for the `pd_aud_max_generation_size` and `pd_aud_async_buff_size` operands.

Effects on individual estimation formulas

If the value of the `pd_aud_sql_source_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining audit trail file capacity*

60) pd_aud_sql_data_size = *size-of-sql-data-output-to-audit-trail*

~<unsigned integer>((0-1000000))(bytes)

Specifies the size in bytes of the SQL data output to the audit trail when using the security audit facility. When `0` is specified, no SQL data is output to the audit trail. For SQL data larger than the specified value, the portion in excess of the specified value is not output to the audit trail.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

Relationship to other operands

If you specify this operand, re-estimate the specifications for the `pd_aud_max_generation_size` and `pd_aud_async_buff_size` operands.

Effects on individual estimation formulas

If the value of the `pd_aud_sql_data_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining audit trail file capacity*

3.2.20 Operands related to the system switchover facility

61) pd_ha_acttype = <u>monitor</u> | server

Specifies whether to run the system switchover facility in the monitor mode or server mode. For details about the monitor and server modes, see the *HiRDB Version 9 System Operation Guide*.

`monitor`: The system switchover facility runs in the monitor mode.

`server`: The system switchover facility runs in the server mode.

If any of the following facilities is used, the server mode is used as the default.

- User server hot standby
- Rapid system switchover facility
- Standby-less system switchover (1:1) facility
- Standby-less system switchover (effects distributed) facility

Conditions

To specify this operand, you must specify `use` for the `pd_ha` operand, and `N` for the `pd_ha_ipaddr_inherit` operand. If you specify `server`, all of the following conditions must be satisfied:

- Hitachi HA Toolkit Extension is installed.
- It is specified that IP addresses are not inherited. (`pd_ha_ipaddr_inherit = N` is specified.)

If you specify `server` without installing these products, HiRDB cannot start.

62) pd_ha_unit = nouse

Specify this operand if the system switchover facility is not to be applied to the unit. Specify this operand when the facility is to be applied to only some of the units. Specifying this operand invalidates the specifications of all of the operands explained in this section.

Relationship to other operands

If the `-k stls` option is specified in the front-end server's `pdstart` operand, a recovery-unnecessary front-end server unit is assumed. The system switchover facility is not applicable to a recovery-unnecessary front-end server unit. For a system that employs the system switchover facility, make sure that you specify this operand in the unit control information definition for the recovery-unnecessary front-end server unit.

63) pd_ha_switch_timeout = Y | N

This operand can be specified when the server mode is used. Specification of this operand is invalid in the monitor mode.

This operand specifies whether to switch the system without waiting for internal termination processing of HiRDB when internal termination processing of HiRDB (or a unit for a HiRDB/Parallel Server) during system switchover has exceeded the server failure monitoring time. The server failure monitoring time referred to here is the time specified for the `patrol` operand of the Hitachi HA Toolkit Extension.

For details about the `patrol` operand of Hitachi HA Toolkit Extension, see the manual *Hitachi HA Toolkit*.

`Y`:

Switches the system without waiting for internal termination processing of HiRDB when internal termination processing of HiRDB during system switchover has exceeded the server failure monitoring time. In this case, system switchover is carried out by assuming that HiRDB has slowed down.

If you are using the standby-less system switchover (1:1) facility or the standby-less system switchover (effects distributed) facility, the specification for this operand is invalid during planned system switchover.

`N`:

Does not switch the system until internal termination processing of HiRDB during system switchover is terminated.

Advantage

If internal termination processing of HiRDB during system switchover takes a long time, because, for example, of a disk error, system switchover might be delayed as a result. If you specify `Y` (default value) for this operand, you can switch the system without waiting for internal termination processing of HiRDB, even when it is taking a long time.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `Y`.

Notes

- If you specify `Y` for this operand when the `patrol` operand value is small, planned system switchover might turn into system switchover based on slow-down. This is because internal termination processing of HiRDB during planned system switchover exceeds the time specified by the `patrol` operand.
- You need to be careful if `restart` is specified for the `switchtype` operand of the Hitachi HA Toolkit Extension. If `pd_ha_switch_timeout=Y` (default value) is specified, and if internal termination processing of HiRDB exceeds the server failure monitoring time, HiRDB is not started in the system in which the failure occurred. In this case, the system is immediately switched.

  For details about the `switchtype` operand of the Hitachi HA Toolkit Extension, see the manual *Hitachi HA Toolkit*.

64) pd_ha_server_process_standby = Y | N

Specifies whether to use user server hot standby. For details about user server hot standby, see the *HiRDB Version 9 System Operation Guide*.

`Y`: Uses user server hot standby.

`N`: Does not use user server hot standby.

Omit this operand if you use the standby-less system switchover (1:1) facility or standby-less system switchover (effects distributed) facility.

Condition

HiRDB must be in the server mode if you are to specify `Y` (default value) for this operand. If the monitor mode is in effect, `N` is always assumed.

Relationship to other operands

The rapid system switchover facility includes user server hot standby. Therefore, if `standbyunit` is specified (to use the rapid system switchover facility) for the `pd_ha_agent` operand, `Y` is assumed for the `pd_ha_server_process_standby` operand, even if you specify `N` for it.

65) pd_ha_agent = standbyunit | server | activeunits

Specify this operand when you use the following facilities:

- Rapid system switchover facility

- Standby-less system switchover (1:1) facility
- Standby-less system switchover (effects distributed) facility

For details about these facilities, see the *HiRDB Version 9 System Operation Guide*.

`standbyunit`: Specify this option when you use the rapid system switchover facility.

`server`: Specify this option when you use the standby-less system switchover (1:1) facility.

`activeunits`: Specify this option when you use the standby-less system switchover (effects distributed) facility.

Note that you cannot specify both the standby-less system switchover (1:1) facility and the standby-less system switchover (effects distributed) facility within the same system.

Condition

If you specify this operand, the server mode must be used. If the monitor mode is used, the error message `KFPS01896-E` is output.

Notes

- A unit that is the target of the rapid system switchover facility cannot inherit IP addresses. Therefore, for a HiRDB/Single Server, specify `N` for the `pd_ha_ipaddr_inherit` operand in the system common definition or unit control information definition. If a configuration to inherit IP addresses is used for a HiRDB/Parallel Server, specify `N` for the `pd_ha_ipaddr_inherit` operand in the unit control information definition of the unit that is the target of the rapid system switchover facility.
- When you use the rapid system switchover facility, standby-less system switchover (1:1) facility, or standby-less system switchover (effects distributed) facility, global buffers cannot be dynamically changed. For details about how to dynamically change global buffers, see the *HiRDB Version 9 System Operation Guide*.

Relationship to other operands

- The rapid system switchover facility includes user server hot standby. Therefore, if you specify `standbyunit` for this operand, `Y` is assumed for the `pd_ha_server_process_standby` operand.
- The standby-less system switchover (1:1) facility and the standby-less system switchover (effects distributed) facility use the server process of the unit at the switching destination following a system switchover. Consequently, if you specify `server` or `activeunits` for this operand, the specification of the `pd_ha_server_process_standby` operand becomes invalid.
- When you use the rapid system switchover facility, standby-less system switchover (1:1) facility, or standby-less system switchover (effects distributed) facility, `Y` is assumed for the `pd_rdarea_open_attribute_use` operand (RDAREA opening trigger specification).
- When you use the standby-less system switchover (effects distributed) facility, not all units within an HA group need to be active, and therefore the specification of the `pd_start_skip_unit` operand becomes invalid. However, all back-end servers within the HA group must be running at some units. Otherwise, HiRDB cannot start.
- When you specify this operand, check the values specified for the following operands. If the specification values are wrong, the error message `KFPS01896-E` is output.

Operand name	pd_ha_agent operand value		
	standbyunit	server	activeunits
`pd_ha`	`use`	`use`	`use`
`pd_ha_acttype`	`server`	`server`	`server`
`pd_ha_ipaddr_inherit` (unit control information definition)	`N`	--	--
`pd_ha_agent`	--	`activeunits` must not be specified for other units.	`server` must not be specified for other units.
`-c` option of the `pdstart` operand	--	Y	--
`-g` option of the `pdstart` operand	--	--	Y

Operand name	pd_ha_agent operand value		
	standbyunit	server	activeunits
`pdhagroup`	--	--	Y

Legend:

Y: Operand must be specified.

--: Not applicable

Effects on individual estimation formulas

If the value of the `pd_ha_agent` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Parallel Server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

66) pd_ha_max_act_guest_servers = *maximum-number-of-acceptable-guest-BES*

~<unsigned integer>((0-33))

Specify this operand when you use the standby-less system switchover (effects distributed) facility. This operand specifies the maximum number of guest back-end servers that operate as running systems inside the unit. When the number of guest back-end servers that are operating as running systems inside the unit reaches the value specified for this operand, the accepting state for guest back-end servers is cancelled.

The value of this operand determines the guest area to be allocated by the accepting unit. Adjust the resource requirement for back-end servers by changing the value of this operand.

If the value of this operand is too small, processing for some of the back-end servers might not take place when a system switchover occurs. When the number of back-end servers in an error state inside the HA group exceeds the total number of acceptable guest back-end servers inside the HA group, processing for the excess back-end servers is not accepted by any unit, and consequently, processing for these back-end servers does not occur.

Condition

When you specify this operand, you must also specify the following operands. If they are not specified, a system definition error occurs (message `KFPS01896-E` is output).

- `pdhagroup`
- `pd_ha_agent = activeunits`

Specification guidelines

If this operand is omitted, HiRDB calculates a value for this operand. The upper limit of the value that can be specified is the smaller of the following two values:

- 34 - *number-of-host-back-end-servers-inside-unit*
- *number-of-servers-inside-HA-group* - *number-of-host-BESs-inside-unit*

Even if you specify a value greater than the upper limit, it is corrected to the upper limit (message `KFPS05613-W` is output).

Note that the maximum number of acceptable guest back-end servers calculated by HiRDB when this operand is omitted assumes that all of the following conditions are satisfied:

- All units inside the HA group have the same number of back-end servers.
- A 2-point error has occurred.

Therefore, if all units inside the HA group do not have the same number of back-end servers, or if a 3-point error occurs, the default value of this operand might not allow processing in some of the back-end servers.

Operand default

The following formula can be used to calculate the default value for this operand:

- When the number of units inside the HA group is 2: *a*
- When the number of units inside the HA group is 3 or larger: $\uparrow (a \times 2) \div (b - 2) \uparrow$

a: *number-of-back-end-servers-inside-host*

b: *number-of-units-inside-HA-group*

When you specify this operand, use the calculation formula (recommended value) shown below to determine the value to be specified for this operand. This calculation formula assumes that when a multi-point error occurs sequentially starting with the unit containing the largest number of units, except the local unit, the remaining units accept the host back-end servers from the failed unit evenly.

$$\text{Recommended value} = \uparrow \sum_{i=1}^{c} a_i \div (b - c) \uparrow$$

a: *number-of-host-back-end servers-in-the-unit-ranked-i-th-in-number-of-host-back-end servers-excluding-local-unit*

b: *number-of-units-inside-HA-group*

c: *number-of-multi-point-errors-assumed* (< *b*)

Effects on individual estimation formulas

If the value of the `pd_ha_max_act_guest_servers` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

67) pd_ha_max_server_process = *maximum-number-of-user-server-processes-inside-accepting-unit*

~<unsigned integer>((1-10000)) <<*total-value-of-the-pd_max_bes_process-operands-of-host-back-end-servers-inside-unit*>>

Specify this operand when you use the standby-less system switchover (effects distributed) facility. Specify the maximum number of user server processes to be started inside a unit. In the accepting unit, the number of server processes is restricted by the value of the `pd_max_bes_process` operand (the maximum number of processes that can be started) for each host BES and guest BES. In addition, specifying this operand can restrict the total number of server processes inside the unit.

Condition

If you specify this operand, you must also specify the following operands. If they are not specified, the specification of the `pd_ha_max_server_process` operand becomes invalid.

- `pdhagroup`
- `pd_ha_agent = activeunits`

Specification guidelines

Omit this operand if you do not wish to increase the number of processes inside the unit following a system switchover. However, the upper limit of the number of service requests that can be concurrently processed might become restricted. If there is a sufficient amount of resources, specify the combined total value of the maximum number of processes that can be started (value of the `pd_max_bes_process` operand) in the host back-end servers inside the unit and the guest back-end servers that can be accepted. Then, the upper limit of the number of service requests that can be concurrently processed stays the same after a system switchover.

Specifying this operand can prevent an excessive load from being applied to the accepting unit. Determine the value to be specified for this operand by taking into account both the load increase on the unit and the number of service requests that can be concurrently processed following a system switchover.

Note

If the value specified for this operand is smaller than the combined total value of the `pd_max_bes_process` operand of the host back-end servers inside the unit, that specified value is invalid. In this case, the combined total value of the `pd_max_bes_process` operand of the host back-end servers inside the unit is assumed (message `KFPS05614-W` is output).

68) pd_ha_process_count = *number-of-processes-resident-inside-unit-after-acceptance-of-guest-BES*

~<unsigned integer>((0-10000))

Specify this operand when you use the standby-less system switchover (effects distributed) facility. Specify the combined total of the number of host back-end servers inside the unit and the number of resident processes for the guest back-end servers after these guest back-end servers have been accepted.

You can use the specification for this operand to adjust the number of resident processes in each server after guest back-end servers are accepted.

Condition

If you specify this operand, you must also specify the following operands. If they are not specified, the specification of the `pd_ha_max_server_process` operand becomes invalid.

- `pdhagroup`
- `pd_ha_agent = activeunits`

Specification guidelines

- If you omit this operand, HiRDB calculates a value for this operand.
- If some back-end servers cannot start a process, and service requests cannot be processed because a resident process not currently in service is occupying the process, decrease the value of this operand. However, transaction throughput might deteriorate following a system switchover. To improve the transaction throughput following a system switchover, increase the value of this operand. However, if the value of this operand is too large, some of the back-end servers might not be able to process service requests.

Note

If the value specified for this operand is smaller than the combined total value of the `pd_process_count` operand of the host back-end servers inside the unit, that specified value is invalid. In this case, the combined total value of the `pd_process_count` operand of the host back-end servers inside the unit is assumed (message `KFPS05615-W` is output).

Remarks

The number of processes resident in each back-end server after guest back-end servers are accepted is the smaller of the following two values:

- Value obtained by proportionately distributing the value of this operand based on the value of the `pd_process_count` operand
- Value of the `pd_process_count` operand of each back-end server

$$\mathrm{MIN}\left\{\begin{array}{l}\texttt{pd\_process\_count,}\\[1em] \downarrow \texttt{pd\_ha\_process\_count} \times \dfrac{\texttt{pd\_process\_count}}{\displaystyle\sum_{\text{Executing BES}} \texttt{pd\_process\_count}} \downarrow\end{array}\right\}$$

The default value of this operand is the value obtained by multiplying the maximum number of user server processes that can be started after guest back-end servers are accepted by the ratio between the number of resident processes and the maximum number of user server processes that can be started before guest back-end servers are accepted. The calculation formula follows.

$$\uparrow \texttt{pd\_ha\_max\_server\_process} \times \frac{\displaystyle\sum_{\text{Host BES}} \texttt{pd\_process\_count}}{\displaystyle\sum_{\text{Host BES}} \texttt{pd\_max\_bes\_process}} \uparrow$$

69) pd_ha_resource_act_wait_time = *maximum-wait-time-for-resource-activation*

~<unsigned integer>((2-3600)) (seconds)

This operand specifies the maximum wait time for the running server's resources to be activated at the time of unit startup when you use the standby-less system switchover (effects distributed) facility. Unit startup processing is placed on hold up to the specified amount of wait time. If the resources are activated within the specified amount of time, unit startup processing resumes.

Advantages

When unit startup processing is completed, jobs can be started only if the running server's startup processing is completed in the unit. By specifying an appropriate value in this operand, you can start your jobs immediately after the unit's startup processing is completed because the unit startup processing will have waited on wait status for resource activation.

Specification guidelines

Normally, there is no need to specify this operand. Specify this operand when all the following conditions are applicable:

- The standby-less system switchover (effects distributed) facility is being used.

- The KFPS05623-I message was displayed.
- The target unit for the message contains the running server.

Use the following guideline to determine the value to be specified in this operand:

10 + time required for resource activation processing (seconds)

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is 10.

Remarks

If the unit does not contain the running server, the running server's startup is placed on hold for the amount of time specified in this operand. However, if all the servers in the unit start as standby servers, unit startup processing is restarted without waiting for the amount of time specified in this operand.

70) pd_ha_ipaddr_inherit = Y | N

This operand specifies whether to inherit IP addresses when using the rapid system switchover facility. Decide whether to specify this operand for a HiRDB/Parallel Server. How this facility is run differs depending on whether IP addresses are inherited. For details, see the *HiRDB Version 9 System Operation Guide*.

Y: Inherits IP addresses.

N: Does not inherit IP addresses.

In server mode operations, IP addresses cannot be inherited.

Omit this operand when you use the standby-less system switchover (1:1) facility or the standby-less system switchover (effects distributed) facility. This operand will be ignored if specified in this case.

Specification guidelines

- If a configuration in which IP addresses are inherited is used for a HiRDB/Parallel Server, specify N (no IP address inheriting) only for the units that are the targets of the server mode.
  You cannot specify Y for this operand if you specify N for the pd_ha_ipaddr_inherit operand of the system common definition.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is Y.

3.2.21 Operands related to HiRDB Datareplicator

71) pd_rpl_hdepath = *extracted-side-HiRDB-Datareplicator-directory*

Specifies the extracted side HiRDB Datareplicator directory when the HiRDB Datareplicator linkage facility is used. Specify the name specified in the HDEPATH environment variable of the extracted side HiRDB Datareplicator.

Condition

This specification value is valid when the HiRDB Datareplicator linkage facility is being used and either Y is specified for the pd_rpl_init_start operand or the pdrplstart command is entered.

Note

- To change the extracted side HiRDB Datareplicator directory name, it is necessary to change the HDEPATH environment variable of the extracted side HiRDB Datareplicator.

Relationship to other operands

If you specify the pd_rpl_reflect_mode operand, make sure that you also specify this operand.

3.2.22 Operands related to communication processing

72) pd_service_port = *scheduler-process-port-number*
~<unsigned integer>((5001-65535))

Specifies the port number for the scheduler process under the following circumstances:

- A high-speed connection facility is used

For details about high-speed connection facilities, see the *HiRDB Version 9 Installation and Design Guide.*

- A firewall or NAT is installed on the HiRDB server

  For details about setting the HiRDB environment when a firewall or NAT is installed on the HiRDB server, see the *HiRDB Version 9 Installation and Design Guide*.

Specification guidelines

- Use this operand for a HiRDB/Single Server or when there is a firewall between the HiRDB client and HiRDB server.
- Use this operand in systems connected via a broadband LAN to reduce the number of communications when connecting to a server.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed.

Notes

- If there are multiple units in a single server machine, such as in a mutual system switchover configuration, always specify this operand in the unit control information definition. Specify a different port number for each unit. If a port number is duplicated, system switchover will fail in one of the units during a switching attempt.
- When this operand is specified in both the unit control information definition and the system common definition, the specification in the unit control information definition is effective.
- If the HiRDB port number (the port number specified with the `pd_name_port` operand or the `-p` option of the `pdunit` operand) is specified, the specification of this operand will be ignored, which will prevent operation using the high-speed connection facility or when a firewall or NAT is set up at the HiRDB server. The `KFPS00860-W` message will be issued when this happens.
- If the high-speed connection facility is used to issue concurrently more connection requests than the value set in the `pd_max_users` operand, a shortage will occur in the number of active front-end servers and single servers for extracting connection requests from the message queue. Consequently, some connection requests will not be extracted from the message queue, and the message queue monitoring facility could stop the unit. You must ensure that the number of concurrent connection requests generated does not exceed the value specified in the `pd_max_users` operand. For details about the message queue monitoring facility, see the *HiRDB Version 9 System Operation Guide*.
- For details about how to specify a scheduler process port number and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.
- In the case of a parallel server that has multiple front-end servers, the front-end servers to be connected are fixed and load sharing will no longer be possible. For this reason, consider carefully the balance among the front-end servers to be connected.

Relationship to other operands

This operand is related to the following operands:

- `pd_scd_port`
- `pdunit -s`

73) pd_change_clt_ipaddr = 0 | 1

Specifies the network to be used by the HiRDB server to communicate with HiRDB clients. Normally, you need not specify this operand.

`0`:

Communication from the HiRDB server to HiRDB clients uses the network in which the IP address specified in the `PDCLTRCVADDR` operand of the client environment definition is located. If the `PDCLTRCVADDR` operand is omitted, the IP address of the standard host is assumed.

`1`:

For communication from the HiRDB server to HiRDB clients, the network used for communication from HiRDB clients to the HiRDB server is used.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `1` (If `v6compatible` or `v7compatible` is specified for the `pd_sysdef_default_option` operand, `0` is assumed).

74) pd_registered_port =*"port-number-reservation-range"* [,*"port-number-reservation-range"*]...

~<character string>

When the HiRDB reserved port facility is used, this operand specifies ranges of port numbers to be used for communication by HiRDB.

The HiRDB reserved port facility is enabled for server-to-server communication and server-to-client communications. For details, see the description of the `pd_registered_port_level` operand. This facility does not need to be used if the number of ports to be used is small.

Operand specification method

This example provides the following 35,000 port numbers: 6000 to 8999, 12500 to 29999, and 30500 to 44999:

```
set pd_registered_port = "6000:8999","12500:29999","30500:44999"
```

Advantages

The port numbers to be used for communication between HiRDB servers are assigned automatically by the OS. If the communication volume increases significantly, a shortage of port numbers might cause an interruption of processing or might adversely affect the communication processing by other programs. Use this operand to specify a range of port numbers to be used exclusively by HiRDB to prevent such problems.

Specification guidelines

- The range of port numbers that can be specified is from 5001 through 49151.
- For the number of ports used by HiRDB, see the *HiRDB Version 9 Installation and Design Guide*.
- Do not register the port number specified in this operand in `%windir%\system32\drivers\etc\services` (definition location in a DNS environment). A duplication check is performed for `%windir%\system32\drivers\etc\services` when `Y`, `C`, or `W` is specified in the `pd_registered_port_check` operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed.

Specification value tuning method

As a guideline, the value obtained from the following formula is the total number of ports that will be needed:

$a + b + 100$

a: Number of HiRDB reserved ports that are being used (`# OF REGISTERED PORTS`)

b: Number of ports being used that were assigned automatically by the OS because all HiRDB reserved ports were in use (`# OF ASSIGNED PORTS`)

These values can be determined from the statistical information related to system operation obtained by executing the statistics analysis utility.

Operand rules

- Up to 10 ranges of port numbers can be specified.
- If more than one range is specified, ensure that there is no overlap in the port numbers included in the various ranges.
- An ending port number must be greater than the paired starting port number.

Notes

- When there are multiple units in a single server machine, such as in a mutual system switchover configuration, this operand must be specified in the unit control information definition. Be sure to specify a different port number for each unit. If a port number is duplicated, system switchover will fail in one of the units during a switching attempt.
- Specification of a port number that is not supported by the HiRDB reserved port facility is ignored. For the scope of the HiRDB reserved port facility, see the `pd_registered_port_level` operand.
- For details about how to specify port numbers when the HiRDB reserved port facility is used and for notes about duplication of other port numbers, see the *HiRDB Version 9 Installation and Design Guide*.

- When this operand is specified in both the unit control information definition and the system common definition, the specification in the unit control information definition is effective.

Effects on individual estimation formulas

If the value of the `pd_registered_port` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

75) pd_registered_port_check = Y | N | C | W

Specifies whether checking is to be performed for overlapping port numbers in the ranges of port numbers specified in the `pd_registered_port` operand and in the port numbers registered in `%windir%\system32\drivers\etc\services` (definition location in a DNS environment).

`Y`:

Check for overlap. If an overlap is found, the `KFPS00348-E` message is output and HiRDB activation is canceled.

`N`:

Do not check for overlap.

`C`:

Check for overlap. The HiRDB reserved port facility is not applied to any overlapping port number.

`W`:

Check for overlap. If an overlap is found, the `KFPS00354-W` message is output. The HiRDB reserved port facility is not applied to any overlapping port number.

Condition

The `pd_registered_port` operand must be specified.

Specification guidelines

- If an overlap in port numbers is detected, HiRDB communication might be adversely affected, resulting, for example, in the receipt of wrong messages or message send failures.
- If `Y`, `C`, or `W` is specified, process server process activation in a DNS environment might slow down.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `Y`.

76) pd_registered_port_level = <u>0</u> | 1

Specifies the target range for the HiRDB reserved port facility.

Condition

The `pd_registered_port` operand must be specified.

Specification guidelines

The following shows the target range depending on the value of this operand (`0` or `1`):

Port numbers used by HiRDB		Value of pd_registered_port_level	
		0	1
Ports used by HiRDB server	Send port number for server-to-server communication	Y	Y
	Receive port number for server-to-server communication	Y	Y
	Port number specified in `pd_name_port` operand	N	N
	Port number specified in `-p` option of `pdunit` operand	N	N
	Port number specified in `pd_service_port` operand	N	N

Port numbers used by HiRDB		Value of pd_registered_port_level	
		0	1
	Port number specified in `pd_scd_port` operand	N	N
	Port number specified in `-s` option of `pdunit` operand	N	N
	Port number specified in `pd_trn_port` operand	N	N
	Port number specified in `-t` option of `pdunit` operand	N	N
	Port number specified in `pd_mlg_port` operand	N	N
	Port number specified in `-m` option of `pdunit` operand	N	N
	Port number specified in `pd_alv_port` operand	N	N
	Port number specified in `-a` option of `pdunit` operand	N	N
	Send port number for command	N	N
	Receive port number for command	N	N
	Send port number for client	N	Y
	Receive port number for client	Y	Y
	Service port number for communication port	N	N
	Port number used for lock control	N	N
Ports used by HiRDB client	Send port number	N	N
	Receive port number	N	N

Legend:

Y: Subject to the HiRDB reserved port facility. A port number specified in the `pd_registered_port` operand is used.

N: Not subject to the HiRDB reserved port facility.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

77) pd_ipc_send_retrycount = *process-to-process-send-retries-count*

~<unsigned integer>((1-32767))

Specifies the number of times process-to-process communication can be attempted. This operand is related to the `pd_ipc_send_retrysleeptime` operand.

Examples

- `pd_ipc_send_retrycount = 500`
- `pd_ipc_send_retrysleeptime = 2`

When the operands are specified in this way, send is attempted up to 500 times and a 2-second sleep occurs between attempts.

Specification guidelines

- Normally, this operand need not be specified.
- Specifying too large a value for this operand increases the CPU lock rate.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `200`.

78) pd_ipc_send_retrysleeptime = *process-to-process-send-retry-sleep-time*

~<unsigned integer>((0-60)) (seconds)

Specifies the sleep time between process-to-process send retries.

This operand is related to the `pd_ipc_send_retrycount` operand.

Examples

- `pd_ipc_send_retrycount = 500`
- `pd_ipc_send_retrysleeptime = 2`

When the operands are specified in this way, send is attempted up to 500 times and a 2-second sleep occurs between attempts.

Specification guidelines

- Normally, this operand need not be specified.
- Specifying too large a value for this operand increases communication completion time.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

79) pd_ipc_send_count = *server-to-server-send-retries-count*

~<unsigned integer>((1-32767))

Specifies the number of times a server-to-server send can be performed before the send is completed. A send occurs for up to 5 seconds at each retry. With the default value, send will be performed for up to 60 $\times$ 5 seconds = 300 seconds.

Specification guideline

Normally, this operand need not be specified. If transmission timeouts occur frequently, increase this operand's value.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `60`.

80) pd_ipc_recv_count = *server-to-server-receive-retries-count*

~<unsigned integer>((1-32767))

Specifies the number of times a server-to-server receive can be performed before receive is completed. Receive occurs for up to 5 seconds at each retry. With the default value, receive is performed for up to 120 $\times$ 5 seconds = 10 minutes.

Specification guideline

Normally, this operand need not be specified.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `120`.

81) pd_inet_unix_bufmode = os | conf

Specifies the size of the send/receive buffer to be used for communication with HiRDB.

`os`**:** Use the value set by the OS.

If a send/receive buffer size is specified in any of the following operands, that value is ignored and the value set by the OS is used:

- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand

`conf`: Use the value specified for the following operands:

- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand

Specification guidelines

- Normally, specify `os` in this operand to use the value set by the OS.
- There are advantages and disadvantages of specifying `conf` as described below. If you specify `conf`, perform a thorough evaluation, including a verification of the machine's performance capabilities.

● If the send/receive buffer size is small and large data, such as `BLOB` data, is sent and received, data resend operations occur. This might adversely affect the communication performance and CPU usage rate. In this case, you can improve the communication performance and reduce the CPU usage rate by increasing the send/receive buffer size.

● If the send/receive buffer size is large and small data, such as numeric data, is sent and received, `ACK` might not be returned quickly (delayed `ACK`) when data is received depending on how TCP/IP is configured. This might result in poor performance.

Notes

The send/receive buffer size, OS settings, how to change the size, and the supported size range depend on the OS. For details, see the OS documentation.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `conf` is assumed.

Relationship to other operands

This operand is related to the following operands:

- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

For details about the send/receive buffer size that depends on the combination of the value specified for this operand and the values of the operands related to this operand, see *Relationship to other operands* under the *pd_inet_unix_bufmode* operand in the system common definition.

82) pd_ipc_inet_bufsize = *send-receive-buffer-size-for-server-to-server-communication*

~<unsigned integer>((0-262144)) (bytes)

Specifies, as a multiple of 4,096 bytes, the maximum size of the send/receive buffer to be used for server-to-server communication (TCP INET domain).

Specification guidelines

- Normally, there is no need to specify this operand. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- If you specify `conf` in for the `pd_inet_unix_bufmode` operand, use this operand to specify the send/receive buffer size used for HiRDB communication processing.

Notes

- If `0` is specified, the value set by the OS is used.
- The maximum TCP buffer size depends on the OS.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `16384`.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_tcp_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

83) pd_tcp_inet_bufsize = *send-receive-buffer-size-for-communication-with-HiRDB-client*

~<unsigned integer>((0-262144)) (bytes)

Specifies, as a multiple of 4,096 bytes, the maximum size for the send/receive buffer to be used for communication (TCP INET domain) with the HiRDB client.

Specification guidelines

- Normally, this operand does not need to be specified. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- If you specify `conf` in for the `pd_inet_unix_bufmode` operand, use this operand to specify the send/receive buffer size used for HiRDB communication processing.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `0`.

Notes

- If `0` is specified, the value set by the OS is used.
- The maximum TCP buffer size depends on the OS.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_ipc_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

84) pd_listen_socket_bufset = 0 | 1

Specifies whether the send/receive buffer is to be set in a `listen` socket when the `listen` socket for a TCP INET domain that is used for HiRDB communication processing is created.

`0`: Do not set the send/receive buffer size in the `listen` socket.

`1`: Set the send/receive buffer size in the `listen` socket.

For the send/receive buffer size, the value specified for the `pd_ipc_inet_bufsize` operand is set. However, the following conditions must be satisfied:

- `conf` is specified for the `pd_inet_unix_bufmode` operand.
- A non-zero value is specified for the `pd_ipc_inet_bufsize` operand.

For details, see the description of the *pd_inet_unix_bufmode* operand.

Specification guidelines

- Normally, this operand does not need to be specified. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- When `conf` is specified for the `pd_inet_unix_bufmode` operand, and the send/receive buffer size specified for this operand is used, if the `pd_ipc_inet_bufsize` operand value is smaller than the value set by the OS, a communication delay might occur. In this case, you might be able to improve the communication performance by specifying `1` for this operand.

Notes

If you have specified `1` for this operand, use the send/receive buffer size set by this operand even when you specify `conf` for the `pd_utl_inet_unix_bufmode` operand and perform communication using the `pd_utl_buff_size` operand. Set the send/receive buffer size in such a manner that the same size is used for the `listen` and `accept` sockets. If different send/receive buffer sizes are used for the `listen` and `accept` sockets, the communication performance might not be improved.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `0` is assumed.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_tcp_inet_bufsize` operand

- `pd_ipc_inet_bufsize` operand
- `pd_inet_unix_utl_bufmode` operand
- `pd_utl_buff_size` operand

85) pd_inet_unix_utl_bufmode = auto | conf

Specifies the size of the send/receive buffer to be used for communication that uses the `pd_utl_buff_size` operand.

`auto`:

Automatically calculate the value based on the `pd_utl_buff_size` operand.

`conf`:

Use the value specified for the `pd_ipc_inet_bufsize` operand.

Specification guidelines

- Normally, this operand does not need to be specified. Specify `os` in the `pd_inet_unix_bufmode` operand and use the value set by the OS.
- If you specify `conf` for the `pd_inet_unix_bufmode` operand and `1` for the `pd_listen_socket_bufset` operand, specify `conf` for this operand.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `auto` is assumed.

Relationship to other operands

This operand is related to the following operands:

- `pd_inet_unix_bufmode` operand
- `pd_tcp_inet_bufsize` operand
- `pd_ipc_inet_bufsize` operand
- `pd_listen_socket_bufset` operand
- `pd_utl_buff_size` operand

3.2.23 Operands related to Java

Java operands are specified when a Java stored procedure or Java stored function is used. For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide.*

86) pd_java_archive_directory = *"JAR-file-storage-directory"*

~<path name>

Specifies the name of the directory for storing JAR files used by Java stored procedures or Java stored functions.

Notes

- The specified directory must be created before installation of JAR files.
- The JAR file storage directory is used only for storing JAR files.
- Store only the installed JAR files in the JAR file storage directory.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `%PDDIR%\java`.

87) pd_java_classpath = *"Java-class-path"*

~<path name>

Specifies as an absolute path name the class path to be used by a Java virtual machine.

The class contained in the path specified by this operand can be referenced from the Java method, which is executed as the processing procedure of a Java stored procedure or Java stored function.

If a class with the same name exists in the path specified by this operand and in the JAR file specified as an external routine of the Java stored procedure or Java stored function, the path specified by this operand takes precedence.

Operand rules

- Up to 1,024 characters can be used for the path name.
- Path names are not case sensitive.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed.

88) pd_java_runtimepath = "*Java-Runtime-Environment-root-directory*"

~<path name>

Specifies as an absolute path name the root directory of the Java Runtime Environment.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

Notes

Because Java Runtime Environment (JRE) is no longer included, you must add this operand or change its value when you upgrade HiRDB from a version earlier than 07-03 to version 07-03 or later. For notes about upgrading, see *Using Java stored procedures and functions* in the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `%PDDIR%\jre`.

89) pd_java_libpath = "*Java-virtual-machine-library-directory*"

~<path name>

Specifies the directory that stores the library of the Java virtual machine as a relative path name to the Java Runtime Environment root directory (the value of the `pd_java_runtimepath` operand).

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default value is as follows:

OS	Operand's default value
Windows (excluding the following)	`bin\hotspot`, or `bin\client`#
Windows (IPF)	`bin\server`
Windows (x64)	

#

Note that `bin\client` is assumed if `jvm.dll` does not exist in `bin\hotspot`.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

90) pd_java_stdout_file = "*Java-virtual-machine-standard-output-and-standard-error-output-destination-file*"

~<path name>

Specifies as an absolute path name the file to which the standard output and standard error output are to be output in a Java virtual machine.

Specification guidelines

Because the size of the file specified by this operand is extremely large, this operand is not normally specified. It is recommended that this operand be specified during debugging of a Java stored procedure or Java stored function. There is no limit to the size of the file that can be specified by this operand.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the Java virtual machine standard output and standard error output are ignored.

Note

If there are simultaneous writing attempts from multiple processes, their output contents cannot be guaranteed.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

3.2.24 Operands related to external C stored routines

91) pd_c_library_directory = "*C-library-file-storage-directory*"

~<path name>

Specifies the absolute path name of the directory that stores C library files.

Operand rules

- A maximum of 255 characters can be used for the path name.
- Path names are not case sensitive.

Note

When this operand is used, the directory specified here must be created before C library files can be registered.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default is `%PDDIR%\clib`.

3.2.25 Operands related to modifying the directory for work file output

92) pd_tmp_directory = *work-file-output-directory*

~<absolute path name>((3-200))

Specifies the directory to which HiRDB is to output work files. Before you start HiRDB, you must have created the specified directory and granted permission for it to all HiRDB users.

HiRDB determines the directory for work file output based on a preset priority depending on the command being used. For details about the output files for which this operand's directory specification is applicable, see the manual *HiRDB Version 9 Command Reference*.

You can use the `pdcspool` command periodically to delete unneeded work files stored in the specified directory. For details about the `pdcspool` command, see the manual *HiRDB Version 9 Command Reference*.

Condition

Be sure to create the directory specified in this operand before starting HiRDB. Specify full control for everyone as the access permission for the directory.

If no directory with the name specified in this operand is available when HiRDB starts (because, for example, it was deleted by mistake while HiRDB was stopped), HiRDB creates a directory with the specified name and sets the access permission to full control for everyone.

Advantages

- The output destination for work files that HiRDB outputs can be consolidated at a single location other than the HiRDB directory.

- Periodic deletion with the `pdcspool` command of work files that remain after command execution is facilitated.

These capabilities make disk management easier.

Notes

- The root directory (such as `C:\`) cannot be specified in this operand.
- Do not specify any directory under the HiRDB directory.
- Do not specify the name of a directory that is being used by a program other than HiRDB. If you do so, files output by the other program might be deleted when the `pdcspool` command is executed.

  Note that if one of the following directory names is specified, files will not be deleted, even when the `pdcspool` command is executed (in these cases, the files must be deleted manually by the user):

  - Root directory (such as `C:\`)
- If the specified path name is not valid or the user does not have permission to access the specified directory, HiRDB will not be able to access the directory. When this happens, a warning message will be issued and `%PDDIR%\tmp` will be used as the default directory.
- If there is a valid setting in the `TMP` environment variable and this operand is not specified, the default for this operand will be the value set in `TMP`. The size of a path name set in an environment variable is not checked, so a path name in the `TMPDIR` environment variable that is longer than 512 bytes would not be considered invalid.
- If no value is set in the `TMP` environment variable or the setting in `TMPDIR` is invalid, the default specification for this operand is `%PDDIR%\tmp`.

3.2.26 Operands related to shared memory

93) SHMMAX *maximum-shared-memory-segment-size*

~<unsigned integer> (megabytes)

- 32-bit mode: **((6-2047))**
- 64-bit mode: **((6-4194304))**

Specifies, in megabytes, the maximum segment size for the shared memory for global buffer.

HiRDB allocates shared memory segments for global buffers up to the size specified for this operand. If the total size of the global buffers allocated to RDAREAs in the server machine exceeds the value specified for this operand, multiple shared memory segments are allocated.

When HiRDB starts, HiRDB can allocate a maximum of 16 shared memory segments for global buffers per server. You can use the `pdls -d mem` command to obtain information about the shared memory segments used for global buffers. For details about the `pdls -d mem` command, see the manual *HiRDB Version 9 Command Reference*.

Specification guidelines

- For 32-bit mode

  Normally, this operand does not need to be specified. Specify this operand only if HiRDB startup fails with the `KFPH23005-E` message (error code `24`) issued. For details, see the handling of the `KFPH23005-E` message.

  If the `KFPH23005-E` message displays error code `20`, take the appropriate action by referencing the manual *HiRDB Version 9 Messages*.
- For 64-bit mode

  If the size of shared memory for global buffers is 16 gigabytes or less, there is no need to specify this operand.

  If the size of shared memory for global buffers is greater than 16 gigabytes, specify the value that satisfies the following condition:

  SHMMAX operand value × 16 ≥ *size of shared memory for global buffers*

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the default value is as follows:

- 32-bit mode: `200`
- 64-bit mode: `1024`

Operand rule

If the specified value extends beyond the permitted range, the default value is assumed.

Notes

- For `SHMMAX`, specify a value that is larger than the size of the area for managing the shared memory used for the global buffers. If a larger value is not specified, HiRDB cannot be started. For details about the formula for determining the size of the area for managing the shared memory used for global buffers, see *Formula for size of shared memory used by global buffers* in the *HiRDB Version 9 Installation and Design Guide*.
- If the shared memory is to be allocated to a file under the HiRDB directory, but there is not enough free space on the disk containing the HiRDB directory, an insufficient disk capacity error occurs.
- If the shared memory is to be allocated to a page file, but the page file capacity is not sufficient, an insufficient page file capacity error occurs.
- If you use the facility for fixing shared memory pages, specify a value in this operand that is an integral multiple of the page size of a Windows large page. Shared memory pages are fixed in the following cases:
  - The value `fixed` is specified in the `pd_dbbuff_attribute` operand.
  - A value of `1` or greater is specified in the `pd_max_resident_rdarea_no` operand.

  If the specified value is not an integral multiple of the page size, HiRDB increases it to the next integral multiple of the page size.

Relationship to other operands

This operand is related to the following operands:

- `pd_dbbuff_modify`
- `pdbuffer`
- `pd_max_add_dbbuff_no`
- `pd_max_add_dbbuff_shm_no`
- `pd_sysdef_default_option` (If `v6compatible` is specified for HiRDB for 32-bit mode, the default value of the `SHMMAX` operand is `6`.)

Effects on individual estimation formulas

If the value of the `SHMMAX` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Memory required by in-memory data processing* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Single Server*
- *Memory required by in-memory data processing* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

4 Server Common Definition

This chapter explains the operands of the server common definition.

4.1 Operand formats

The server common definition defines information that is common to individual server definitions (front-end server definitions, dictionary server definition, and back-end server definitions). The values defined here become the default values for operands not specified in the individual server definitions. This section explains the formats used to specify the operands of a server common definition.

Note that the numbers in the following table correspond to the numbers assigned to the explanations of the individual operands in *4.2 Operand explanations*.

For users who are creating HiRDB system definitions for the first time

The first step is to determine the values to be specified for the operands shown in boldface type. In principle, HiRDB can be started once the boldface operands have been specified.

No.	Format	Operand category
1	`[set pd_max_bes_process =` *maximum-number-of-activated-processes-per-back-end-server*`]`	Processes
2	`[set pd_max_dic_process =` *maximum-number-of-activated-processes-per-dictionary-server*`]`	
3	`[set pd_process_count =` *resident-processes-count* `[,`*resident-processes-count-at-server-startup*`]]`	
4	`[set pd_server_cleanup_interval =` *interval-for-stopping-nonresident-server-processes*`]`	
5	`[set pd_max_ard_process =` *asynchronous-READ-process-count*`]`	
6	`[set pd_dfw_awt_process =` *number-of-parallel-writes-for-deferred-write-processing*`]`	
7	`[set pd_work_buff_mode = each	pool]`	Work tables
8	`[set pd_work_buff_size =` *work-table-buffer-size*`]`	
9	`[set pd_work_buff_expand_limit =` *work-table-buffer-expansion-limit*`]`	
10	`[set pd_watch_pc_client_time =` *maximum-client-request-wait-time*`]`	System monitoring
11	`[set pd_spd_syncpoint_skip_limit =` *maximum-number-of-skipped-synchronization-point-dumps*`]`	
12	`[set pd_dfw_syncpoint_skip_limit =` *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*`]`	
13	`[set pd_lck_pool_size =` *server-lock-pool-size*`]`	Lock
14	`[set pd_fes_lck_pool_size =` *front-end-server-lock-pool-size*`]`	
15	`[set pd_lck_pool_partition =` *per-server-lock-pool-partition-count*`]`	
16	`[set pd_fes_lck_pool_partition =` *front-end-server-lock-pool-partition-count*`]`	
17	`[set pd_lck_until_disconnect_cnt =` *total-number-of-tables-and-RDAREAs-to-be-locked-per-server-UNTIL-DISCONNECT-specification*`]`	
18	`[set pd_max_open_holdable_cursors =` *maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*`]`	
19	`[set pd_lck_hash_entry =` *lock-pool-hash-entry-count*`]`	
20	`[set pd_dbsync_lck_release_count =` *global-buffer-lock-release-interval-during-synchronization-point-processing*`]`	

No.	Format	Operand category
21	[set pd_sql_object_cache_size = *SQL-object-buffer-size*]#	Buffers
22	[set pd_table_def_cache_size = *table-definition-information-buffer-size*]	
23	[set pd_auth_cache_size = *user-privilege-information-buffer-size*]	
24	[set pd_view_def_cache_size = *view-analysis-information-buffer-size*]	
25	[set pd_type_def_cache_size = *user-defined-type-information-buffer-size*]	
26	[set pd_routine_def_cache_size = *routine-definition-information-buffer-size*]	
27	[set pd_registry_cache_size = *registry-information-buffer-size*]	
28	[set pd_sds_shmpool_size = *single-server-shared-memory-size*]	Shared memory
29	[set pd_dic_shmpool_size = *dictionary-server-shared-memory-size*]	
30	[set pd_bes_shmpool_size = *back-end-server-shared-memory-size*]	
31	[set pd_rpc_trace = Y | N]#	RPC trace information
32	[set pd_rpc_trace_name = "*name-for-RPC-trace-collection-files*"]#	
33	[set pd_rpc_trace_size = *RPC-trace-collection-file-size*]#	
34	[set pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*]#	Troubleshooting information
35	[set pd_module_trace_timer_level = 0 | 10 | 20]#	
36	[set pd_pth_trace_max = *maximum-number-of-stored-communication-traces*]#	
37	[set pd_max_add_dbbuff_no = *maximum-global-buffers-count-for-dynamic-addition*]	Global buffers
38	[set pd_max_add_dbbuff_shm_no = *maximum-shared-memory-segments-count-for-dynamic-addition*]	
39	[set pd_max_temporary_object_no = *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*]#	Temporary tables
40	[set pd_audit_def_buffer_size = *security-audit-information-buffer-length*]	Security
41	[set pd_java_stdout_file = "*Java-virtual-machine-standard-output-and-standard-error-output-destination-file*"]#	Java
42	[set pd_java_castoff = Y | N]	
43	[set pd_log_dual = Y | N]	System log files
44	[set pd_log_remain_space_check = warn | safe]	
45	[set pd_log_auto_unload_restart = Y | N]	
46	[set pd_log_singleoperation = Y | N]	
47	[set pd_log_rerun_reserved_file_open = Y | N]	
48	[set pd_log_rerun_swap = Y | N]	
49	[set pd_log_swap_timeout = *wait-time-for-completion-of-system-log-file-swapping*]	
50	[set pd_log_unload_check = Y | N]	
51	[set pd_log_max_data_size = *log-input/output-buffer-size*]	
52	[set pd_log_write_buff_count = *log-output-buffer-sectors-count*]	

No.	Format	Operand category
53	`[set pd_log_rec_leng =` *system-log-file-record-length*`]`	
54	`[set pd_log_rollback_buff_count =` *rollback-log-input-buffer-sector-count*`]`	
55	`[set pd_log_auto_expand_size =` *extension-amount-per-system-log-file-extension-trigger[,extension-limit]*`]`	
56	`[set pd_spd_dual = Y	`<u>`N`</u>`]`	Synchronization point dump files
57	`[set pd_spd_assurance_msg = `<u>`Y`</u>`	N]`	
58	`[set pd_spd_assurance_count =` *number-of-guaranteed-valid-generations*`]`	
59	`[set pd_spd_reduced_mode =` *reduced-mode-operation-option*`]`	
60	`[set pd_spd_reserved_file_auto_open = Y	`<u>`N`</u>`]`	
61	`[set pd_spd_max_data_size =` *synchronization-point-dump-file-buffer-size*`]`	
62	`[set pd_log_sdinterval =` *system-log-output-volume*`[,`*interval*`]]`	
63	`[set pd_sts_initial_error = `<u>`stop`</u>`	continue	excontinue]`	Server status files (when an error occurs)
64	`[set pd_sts_singleoperation = `<u>`stop`</u>`	continue]`	
65	`[set pd_bes_connection_hold = Y	`<u>`N`</u>`]`	BES connection holding facility
66	`[set pd_bes_conn_hold_trn_interval =` *back-end-server-connection-hold-time*`]`	

#: If this operand is omitted, the value specified in the same operand in the system common definition or unit control information definition is used.

4.2 Operand explanations

4.2.1 Operands related to processes

1) pd_max_bes_process = ***maximum-number-of-activated-processes-per-back-end-server***

~<unsigned integer>((1-2048))<<*pd_max_users value*>>

Specifies the maximum number of processes that can be activated per back-end server. For multi front-end servers, processes that exceed the value specified in the `pd_max_users` can sometimes become concentrated in a single back-end server. The `pd_max_bes_process` operand specifies the maximum number of processes that can be activated per back-end server when that number exceeds the value of the `pd_max_users` operand.

Condition

This operand is applicable when multiple front-end servers are used.

Specification guidelines

- The value determined by the following formula indicates the maximum number of processes that might possibly become concentrated in a single back-end server.

  `pd_max_users` value × *number-of-front-end-servers*

  Specify a value for the `pd_max_bes_process` operand by using the value determined here as the upper limit and taking the degree of process concentration in a single back-end server into consideration. Specifying an unnecessarily large value might cause memory shortage.
- If the value specified is smaller than the `pd_max_users` value, the `pd_max_users` value is assumed as the default, and a warning message (`KFPS01888-W`) is output.
- When more processing requests have been issued than there are back-end server processes that can be started, time will be required to process connection requests from the front-end server to the back-end server.

Tuning the specified value

For details about how to tune the maximum number of processes that can be activated, see the *HiRDB Version 9 System Operation Guide*.

2) pd_max_dic_process = ***maximum-number-of-activated-processes-per-dictionary-server***

~<unsigned integer>((1-2048))<<*pd_max_users value*>>

Specifies the maximum number of processes that can be activated per dictionary server. When multiple front-end servers are used, processes that exceed the value specified in `pd_max_users` can sometimes become concentrated in a single dictionary server. Even when multiple front-end servers are not used, processes that exceed the value specified in `pd_max_users` can become concentrated in a single dictionary server if the number of executions of operation commands related to RDAREAs and global buffers (`pdbufls`, `pddbls`, `pdopen`, `pdclose`, `pdhold`, and `pdrels`) exceeds the value specified in `pd_max_users`.

The `pd_max_dic_process` operand specifies the maximum number of processes that can be activated per dictionary server when that number exceeds the value of the `pd_max_users` operand.

Specification guidelines

- The value determined by the following formula indicates the maximum number of processes that might become concentrated in a single dictionary server:

  `pd_max_users` value × *number-of-front-end-servers*

  Specify a value for the `pd_max_dic_process` operand by using the value determined here as the upper limit and taking the degree of process concentration in a single dictionary server into consideration. Specifying an unnecessarily large value might cause a memory shortage.
- If the execution of operation commands related to RDAREAs or global buffers causes processes to become concentrated in a single dictionary server, the number of operation commands to be concurrently executed becomes the maximum number for this operand.
- If the value specified is smaller than the `pd_max_users` value, the `pd_max_users` value is assumed as the default, and a warning message (`KFPS01888-W`) is output.
- When more processing requests have been issued than there are dictionary server processes that can be started, processing delays will result.

Tuning the specified value

For details about how to tune the maximum number of processes that can be activated, see the *HiRDB Version 9 System Operation Guide*.

3) pd_process_count = ***resident-processes-count*****[,*****resident-processes-count-at-server-startup*****]**

~<unsigned integer>

- For single server: **((0-3000))<<*maximum-process-count-at-startup*>>**
- For back-end server: **((0-2048))<<*maximum-process-count-at-startup*>>**
- For front-end or dictionary server: **((0-2048))<< ↑ *maximum-process-count-at-startup* ÷ 2 ↑ >>**

resident-processes-count

Specifies the number of processes that can be made resident in each server. A resident process is a process that is activated at the time the server is started.

Advantage

By activating the processes used by transactions that can be processed concurrently by each server at the time of system startup and keeping them resident, the process startup time can be reduced even when new transactions are entered. However, HiRDB startup will take longer.

Specification guidelines

- The value to be specified is determined on the basis of the process private area of each server process and the real memory size of the processor. For details about the process private areas of server processes, see the *HiRDB Version 9 Installation and Design Guide*.
- If a multi front-end server configuration is used and if the `pd_max_bes_process` or `pd_max_dic_process` operand is specified in addition, specify for the `pd_process_count` operand a value that satisfies the following conditions:
  `pd_process_count` value ≤ (`pd_max_bes_process` or `pd_max_dic_process` value)
- The value specified for this operand must be no more than the maximum number of processes that can be activated for each server, as shown in the following table:

Server type		Maximum number of processes that can be activated
Single server		`pd_max_users` value
Front-end server	Multiple front-end servers	`pd_max_users` value `+ 1`
	Not multiple front-end servers	`pd_max_users` value
Dictionary server		`pd_max_dic_process`# value
Back-end server		`pd_max_bes_process`# value

#: If the `pd_max_dic_process` or `pd_max_bes_process` operand is omitted, the value of the `pd_max_users` operand is assumed as the default.

Tuning the specified value

For details about how to tune the resident processes count, see the *HiRDB Version 9 System Operation Guide*.

Notes

- Because the number of resident processes has a direct effect on the availability of memory space and on the CPU, specifying an unnecessarily large number might prevent HiRDB from starting or might degrade the server machine's processing performance.
- If more processes than the resident process count are needed, additional processes are dynamically started, up to the maximum processes count allowed. However, depending on the value specified for the `pd_max_server_process` operand, it might not be possible to start all of the processes indicated by the maximum processes count.

Operand default

If this operand is omitted (or `0` is specified for it) and the operands are omitted (or `0` is specified for them) in server definitions as well, the following values are assumed:

- For a single server or back-end server: maximum number of processes that can be activated

- For a front-end server or dictionary server: ↑ *maximum-number-of-processes-that-can-be-activated* ÷ 2 ↑

resident-processes-count-at-server-startup

Specifies the number of processes that can be made resident during HiRDB startup.

It is common for resident processes to be activated during HiRDB startup. If there are many resident processes, the amount of time required for HiRDB startup increases proportionately. For example, it takes approximately 1 second to activate a process on a server machine with a 100-MIPS performance rating.

The differences in processing that result depending on whether a resident processes count at server startup is specified are as follows:

- When there is no specification of a resident processes count at server startup (when, for example, `pd_process_count = 500` is specified)

  All 500 resident processes are activated during HiRDB startup, and HiRDB will not start until they are all activated. In this example, it would take a 100-MIPS server machine about 500 seconds to activate all the resident processes during HiRDB startup.

- When a resident processes count at server startup is specified (when, for example, `pd_process_count = 500,50` is specified)

  Some of the resident processes (50 in this case) are activated during HiRDB startup, and the remaining resident processes are activated after HiRDB startup. HiRDB can be started as long as the specified number of resident processes are activated. In this example, it would take a 100-MIPS server machine about 50 seconds to activate 50 resident processes during HiRDB startup. The remaining 450 resident processes (in this example) would be activated after HiRDB startup.

Advantage

The amount of time required for HiRDB startup is reduced. Use this option when you want to reduce the HiRDB startup time as much as possible, such as when the system switchover facility is being used.

Specification guidelines

Specify a value equal to the number of processes that will be required immediately after HiRDB startup is completed.

Note

When a resident processes count at server startup is specified, recheck the value in the `PDCWAITTIME` operand of the client environment definition.

If more UAPs than the resident processes count at server startup are to be executed immediately following HiRDB startup, transaction processing might not be performed until after the remaining resident processes have been activated. Therefore, if the value specified in the `PDCWAITTIME` operand of the client environment definition is small, it might not be possible to process some UAPs due to timeouts. For details about the `PDCWAITTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.

4) pd_server_cleanup_interval = *interval-for-stopping-nonresident-server-processes*

~<unsigned integer> ((0-1440))<<0>> (minutes)

Specifies in minutes the interval for checking for nonresident server processes in HiRDB that are to be stopped. The facility for stopping nonresident server processes is applied when the number of executing server processes exceeds the number of processes that can be made resident (value specified by the `pd_process_count` operand). The number of server processes that the facility stops is computed automatically by HiRDB.

Advantage

This facility improves the utilization rates of the memory and of process resources because it increases the number of nonresident processes that can be reused when the workload (number of server processes that are executing) peaks.

Specification guidelines

- For example, if there is only one hour a day during which peak workload occurs and the intervals between peaks within that hour are approximately two minutes apart, specify `2` for this operand.
- This facility does not have any effect if the number of server processes that execute concurrently during peak periods is always fewer than the number of resident processes. In this case, omit this operand.

Tuning the specified value

Collect statistical information on system operations for each server for one week. Determine the workload peaks from the number of server processes being serviced (`# OF PROCESSES ON SERVICE`). If that peak value exceeds the currently set number of processes that can be made resident (value specified by the pd_process_count operand), determine the interval between individual peaks and set that number of minutes.

However, if there are ample resources, such as memory and CPU, in the server machine, adding the shortfall in the number of processes to the number of resident processes (in other words, increasing the value of the pd_process_count operand) is more effective in improving performance than specifying the pd_server_cleanup_interval operand.

Note

When this operand is omitted or 0 is specified, the system checks every 10 seconds for nonresident server processes that are waiting to be serviced and stops any such processes that are found.

5) pd_max_ard_process = *asynchronous-READ-process-count*

~<unsigned integer>((0-256))<<0>>

Specify this operand if you use the asynchronous READ facility. For this operand, specify the number of processes necessary for asynchronous READ operations. For a HiRDB/Parallel Server, this operand specifies the number of processes per server (back-end server or dictionary server). For details about the asynchronous READ facility, see the *HiRDB Version 9 Installation and Design Guide*.

Condition

A value of 1 or greater must be specified for the -m option of the pdbuffer operand.

Specification guidelines

- Specify 0 or 1. However, if a value between 2 and 256 is specified for the -m option of the pdbuffer operand, specify the same value as the -m option value. If a value greater than 256 is specified for the -m option, specify the same value as the number of disk devices that store RDAREAs and system files (the number of such disk devices per server for a HiRDB/Parallel Server) or 256.
- Increasing the value of this operand can shorten the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. Decreasing the value of this operand might increase the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. This is because asynchronous READ processes might have to wait for processing completion.
- Because a number of processes equaling *value of this operand* × *server count* are started, determine a value for this operand by taking resources (shared memory and message queue) into consideration. For details about estimating shared memory and message queue sizes, see the *HiRDB Version 9 Installation and Design Guide*.

Tuning the specified value

For details about how to tune the specification value (number of asynchronous READ processes), see the *HiRDB Version 9 System Operation Guide*.

Operand rule

If you specify 0 for this operand, the asynchronous READ facility is not used.

Relationship to other operands

If you change the value of this operand, re-evaluate the value of the pd_max_server_process operand.

Effects on individual estimation formulas

If the value of the pd_max_ard_process operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Processes started by HiRDB/Parallel Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 2* under *Formulas for shared memory used by a single server*

- *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 2* under *Formulas for the size of the shared memory used by a back-end server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

6) pd_dfw_awt_process = *number-of-parallel-writes-for-deferred-write-processing*

~<unsigned integer>((2-255))

Specify this operand when you use the facility for parallel writes in deferred write processing for all buffer pools. Specify for this operand the number of processes to be processed in parallel. Increasing the number of processes can shorten the write processing time. For details about the facility for parallel writes in deferred write processing, see the *HiRDB Version 9 Installation and Design Guide*.

Specification guidelines

Specify `2`, which is the smallest value that enables the facility for parallel writes in deferred write processing. Furthermore, to determine the value for this operand, see *Tuning deferred write processing* in the *HiRDB Version 9 System Operation Guide*.

Note

Specifying the facility for parallel writes in deferred write processing increases the number of processes and consequently raises the CPU usage rate.

Effects on individual estimation formulas

If the value of the `pd_dfw_awt_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Processes started by HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 5* under *Formulas for shared memory used by a single server*
- *Formula 4* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 5* under *Formulas for the size of the shared memory used by a back-end server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

4.2.2 Operands related to work tables

7) pd_work_buff_mode = each | <u>pool</u>

Specifies the method of allocating buffers when HiRDB creates tables.

`each`: Allocate a buffer for each work table.

`pool`: Allocate a buffer pool for each server process.

Specification guidelines

- Normally, `pool` is specified. `pool` (default value) is the appropriate specification when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.
- When the size of the process private area that can be used for work table buffers is predetermined, specify `pool`. When `pool` is specified, HiRDB efficiently allocates work table buffers to work tables.

  In such a case, the process private area is occupied on the basis of the value specified in `pd_work_buff_size`, and input/output operations on work tables are buffered in that pool. Therefore, the process private memory is occupied only to the extent of the value specified in `pd_work_buff_size`.

Notes

If `each` is specified for this operand, the amount of memory used might increase.

Relationship to other operands

If `v6compatible` is specified for the `pd_sysdef_default_option` operand, the default value of this operand is `each`.

Effects on individual estimation formulas

If the value of the `pd_work_buff_mode` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Considerations when migrating to 64-bit mode*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

8) pd_work_buff_size = ***work-table-buffer-size***

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

Specifies in kilobytes the size of buffers for work tables to be created by HiRDB.

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
Advantage	A large work table buffer size reduces the number of I/O operations associated with data manipulation, which means that the execution time of SQL statements that use work tables is also reduced. However, because each server's process private memory is used, you must also take into account the overall size of the system memory (real memory and virtual memory) when you specify this option. If `pd_work_buff_mode = each` is specified, the memory size to be allocated is *value of* `pd_work_buff_size` × *required number of work tables*. Therefore, specifying an unnecessarily large value might cause a virtual memory shortage for other processes.	
Application criterion	Specify `pd_work_buff_mode = pool` when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.	
Specification guidelines	• Specify the size of the buffer to be allocated for one work table. • If a value greater than the work table memory capacity is specified for the work table buffer size, input/output to the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the work table memory capacity: *Work table memory capacity = Applicable work table size*# ÷ 2	• Specify the size of the buffer pool to be allocated for the entire server process. • Specify a value between `4352` and `5120` when a large volume of data is to be retrieved or when manipulations such as join, `ORDER BY`, and `GROUP BY` are to be performed. Specifying such a value increases the unit of sorting input/output, thus reducing the sort time. • If a large value is specified for the work table buffer size, with the total work table memory capacity for each SQL statement as the upper limit, input/output operations on the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the total work table memory capacity for each SQL statement: *Total work table memory capacity per SQL statement = a* × *b + c* × *d*
Notes	• When multiple users execute processes concurrently or when an SQL statement that uses multiple work tables is executed, a buffer of the specified size is allocated for each work table. Consequently, specifying a large value might result in a memory shortage. • If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.	• If the value specified for the work table buffer size is smaller than the number of work tables to be used by each SQL statement, the processing time might become longer than when `each` is specified. Specifically, specify a value that is at least equal to *maximum number of work tables for each SQL statement* × 128. The following formula can be used to determine the maximum number of work tables for each SQL statement: *Maximum number of work tables for each SQL statement = b + d*

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
		• If the specified buffer size is too large to be allocated in the system, the allocation of process private memory fails and the server cannot start up.
Operand rule	Specify a multiple of 128. If the value is not a multiple of 128, it is rounded up to the next multiple of 128.	• Specify a multiple of 128. If the value is not a multiple of 128, it is rounded up to the next multiple of 128. • Specify at least `384`. If a value that is smaller than `384` is specified, it is rounded up to `384`.
Default values	• If both this operand and the `pd_work_buff_size` operand of the single server definition are omitted, 128 is assumed. • If both this operand and the `pd_work_buff_size` operand of the back-end server definition and the dictionary server definition are omitted, 512 is assumed.	• If this operand and the `pd_work_buff_size` operand are both omitted in a single server definition, `384` (in the 32-bit mode) or `5120` (in the 64-bit mode) is assumed. • If this operand and the `pd_work_buff_size` operand are both omitted in a back-end server definition or a dictionary server definition, `1024` (in the 32-bit mode) or `5120` (in the 64-bit mode) is assumed.

a:

$\uparrow$ {Capacity of work table (for storing column information)# (kilobytes) $\div$ 2} $\div$ 128 $\uparrow$ $\times$ 128

b:

Maximum number of work tables (for storing column information)#

c:

$\uparrow$ {Capacity of work table (for storing positional information)# (kilobytes) $\div$ 2} $\div$ 128 $\uparrow$ $\times$ 128

d:

Maximum number of work tables (for storing positional information)#

#: For details about how to determine these values, see the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_work_buff_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

9) pd_work_buff_expand_limit = *work-table-buffer-expansion-limit*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

The size of the work table buffer to be created by HiRDB is specified by the `pd_work_buff_size` operand. Specify the `pd_work_buff_expand_limit` operand if you want to automatically expand a work table buffer when the space in this buffer becomes insufficient. The work table buffer is expanded up to the size specified by this operand.

For example, when the following values are specified for the operands, a 1,024-kilobyte work table buffer is normally allocated. When this size becomes insufficient, the work table buffer is expanded up to 2,048 kilobytes.

- `pd_work_buff_size = 1024`
- `pd_work_buff_expand_limit = 2048`

HiRDB expands a work table buffer in the following cases:

- The necessary work table buffer cannot be allocated when hash execution is applied to an execution method that uses hash join or subquery hash as the joining method.
- A 128-kilobyte work table buffer allocated to each work table becomes insufficient when multiple work tables are concurrently used.

Condition

The pd_work_buff_mode operand must be omitted or pool must be specified for it.

Advantage

You can prevent a work table buffer shortage (too small a value specified for the pd_work_buff_size operand) from causing UAP errors.

Notes

- A work table buffer is not expanded when either of the following conditions is satisfied:
  - The pd_work_buff_expand_limit operand is not specified.
  - pd_work_buff_expand_limit operand value ≤ pd_work_buff_size operand value
- If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.

Operand rule

Specify a multiple of 128. If a value other than a multiple of 128 is specified, it is automatically rounded up to a multiple of 128.

Relationship to other operands

When a work table buffer is expanded for the first time in each server process, the KFPH29008-I message is output. Note that you can use the pd_work_table_option operand to suppress this message output.

Note

After a work table buffer has been expanded, when the number of work tables being used by the applicable server process goes to zero, the expanded work table buffer is released. The number of work tables being used can go to zero in the following cases:

- All cursors that were being used are closed. (In this case, the number of work tables being used might not go to zero.)
- A transaction is normally terminated or cancelled when a holdable cursor is not being used.
- A UAP is disconnected from HiRDB when a holdable cursor is being used.

Remarks

Hash join, subquery hash execution is applied in the following cases:

- *Application of optimizing mode 2 based on cost* and *hash join, subquery hash execution* are specified in the pd_additional_optimize_level operand, the PDADDITIONALOPTLVL operand of the client environment definition, or the ADD OPTIMIZE LEVEL operand of the SQL compile option.
- HASH is specified for the SQL optimization specification of the joining method inside an SQL statement.
- HASH is specified for the SQL optimization specification of the subquery execution method inside an SQL statement.

Effects on individual estimation formulas

If the value of the pd_work_buff_expand_limit operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

4.2.3 Operands related to system monitoring

10) pd_watch_pc_client_time = *maximum-client-request-wait-time*
~<unsigned integer>((0-65535))<<3600>>(seconds)

Specifies in seconds the maximum amount of time for a server to wait for the next request from a HiRDB client after the HiRDB server returns a response to a request from a Windows-compatible HiRDB client.

If no request comes from the HiRDB client within the specified amount of time, it will be assumed that an error occurred at the client and the connection between the server and the client will be terminated forcibly. No notice of disconnection is sent to the HiRDB client in such a case.

The time that is monitored is the period between CONNECT and DISCONNECT (that is, the non-transaction status time), excluding the period between SQL execution startup and COMMIT or ROLLBACK.

Notes

- When 0 is specified, the server waits indefinitely for the next request from the HiRDB client.
- If a small value (up to 600, for example) is specified for this operand, the HiRDB client might detect *server down* during SQL execution and might not terminate correctly.
- For a UNIX edition of a HiRDB client (including a Linux edition of a HiRDB client), time is not monitored, regardless of the value specified for this operand. To monitor time for a UNIX edition of HiRDB client, specify the PDSWATCHTIME client environment definition of the HiRDB client.

Relationship to client environment definition

The value of this operand can be modified for each client. To do so, the PDSWATCHTIME operand must be specified in the client environment definition. For details about the PDSWATCHTIME operand, see the *HiRDB Version 9 UAP Development Guide*.

11) pd_spd_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps*

~<unsigned integer>((0, 2-100000))

If an event such as an endless loop occurs in a UAP, acquisition of synchronization point dumps might be skipped. If several synchronization point dumps are not acquired (instead, they are skipped), there will be an increase in the number of overwrite-disabled system log files, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

If HiRDB is forcibly terminated or terminates abnormally when the number of system log files that cannot be overwritten has reached one-half or more of all system log files, a shortage of system log files occurs during rollback processing when HiRDB is restarted. In this case, HiRDB cannot be restarted unless new system log files are added. Any such restart processing will take longer than normal.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction). When the number of skipped synchronization point dumps reaches the specified operand value, the target transaction is cancelled forcibly and rolled back. This is called the *skipped effective synchronization point dump monitoring facility*.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

Advantage

Specifying this operand provides a means of dealing with endless loops in UAPs.

Specification guidelines

- Normally, specify 0 for this operand. When 0 is specified, HiRDB computes the upper limit for the skip count. If specifying 0 causes a problem or the KFPS02101-I message is issued, change the value of this operand. For guidelines on the value to specify, see the *HiRDB Version 9 System Operation Guide*.
- If the specified value is too large, all system log files might be placed in overwrite disabled status. If this happens, HiRDB terminates abnormally.
- If the specified value is too small, the number of transactions that are forcibly rolled back might increase.
- Specify this value taking into account the value of pd_log_sdinterval and the number of times synchronization point dumps are acquired when the transaction with the longest execution time and largest log output volume is processed.

Operand default value

If this operand is omitted, the skipped effective synchronization point dump monitoring facility is not used.

Notes

- The monitoring provided for by the specification of this operand begins the first time a synchronization point dump is collected after HiRDB is started (including a restart).
- When this operand is specified, UAPs that are being executed in the no-log mode are also monitored. If the processing of a UAP being executed in the no-log mode is cancelled, the database cannot be automatically recovered, and thus RDAREAs are placed in the error shutdown state. Therefore, when specifying the upper limit, account also for the size of the system logs that are output from other transactions inside the server during the processing of UAPs that are executed in the no-log mode.
- If a transaction is delayed due to a high workload, the skip count increases because a synchronization point dump cannot be acquired until the delayed transaction is completed.

- If the skip count exceeds the upper limit, any process executing a transaction is forcibly terminated. In this case, the rollback logs, which are the logs output by these transactions, are output after the forced termination.
- The `pdload`, `pdmod`, `pdrorg`, `pdexp`, `pddbst`, `pdgetcst`, `pdrbal`, `pdvrup`, `pdmemdb`, and `pdextfunc` commands are not monitored by this facility.

12) pd_dfw_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*

~<unsigned integer>((0-100000))<<0>>

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing is completed, acquisition of the synchronization point dump is skipped. This is because acquisition of the synchronization point dump is delayed by the deferred write processing, and the number of update buffers output by the synchronization point exceeds the number of update buffers that can be output within the synchronization point dump acquisition interval.

If several synchronization point dumps are skipped, there will be an increase in the number of system log files that cannot be overwritten, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction) due to deferred write processing.

When the number of skipped synchronization point dumps due to deferred write processing reaches the specified operand value, HiRDB determines the maximum number of update buffers in such a manner that acquisition of a synchronization point dump can be completed within the synchronization point dump acquisition interval. If the maximum number of update buffers is exceeded, HiRDB then outputs the oldest update buffer and limits the total size of update buffers at a synchronization point. This is called the *update buffer size restriction facility*.

Advantage

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing has terminated, the unit terminates abnormally. This operand enables such abnormal termination of the unit to be avoided.

Specification guidelines

Normally, you will omit this operand. If you wish to prevent unit abnormal termination caused by the occurrence of a synchronization point before termination of deferred write processing within the synchronization point dump acquisition interval, specify 1.

If an acceptable number of times synchronization point dumps can be skipped can be determined in advance, such as from the size of the log information, specify that value.

Operand rules

If 0 is specified in this operand, HiRDB does not use the update buffer size restriction facility.

Notes

The following notes explain the period of effectiveness of the update buffer size restriction facility. The period of effectiveness means the interval between issuance of the `KFPH23035-I` message and issuance of the `KFPH23036-I` message.

- If the number of update buffers exceeds the maximum number of update buffers determined by HiRDB, update buffers are output after update processing has executed; this degrades the update processing throughput. You can use the following formula to obtain the maximum number of update buffers determined by HiRDB:

$$\begin{aligned}&(\textit{synchronization point dump interval} \div \textit{unit value of WRITE}^{\#})\\ &\times\ (1 - (\textit{amount of log information from the previous synchronization point dump to the pre-synchronization point})\\ &\times\ (\textit{number of buffer sectors in buffer pool} \div \textit{total number of buffer sectors in buffer pool that was updated at the synchronization point})\end{aligned}$$

#: For details about the unit value of `WRITE`, see the *HiRDB Version 9 System Operation Guide*.

- If the deferred write trigger is specified in the `pd_dbbuff_rate_updpage` or `pdbuffer -y` operand, and each operand value becomes greater than the maximum number of update buffers determined by HiRDB, the maximum number of update buffers determined by HiRDB is changed to the number of update buffers that triggers deferred write processing.

  The value of the `pdbuffer -w` operand is adjusted automatically so that up to the maximum number of update buffers is output for each buffer.

- A skipped synchronization point dump is detected during update buffer output processing at a synchronization point. Therefore, the update buffer size restriction facility might be enabled after a synchronization point dump is skipped and an error message is displayed.
- Normally, when the parallel writes facility is used, there is one output request per synchronization point for each parallel WRITE process for deferred write processing during synchronization point processing. However, if the update buffer restriction facility is used, there will be more than one output request in order to facilitate early detection of skipped synchronization point dumps.

4.2.4 Operands related to lock

13) pd_lck_pool_size = ***server-lock-pool-size***

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-2000000))<<16000>>**
- 64-bit mode: **((1-2000000000))<<32000>>**

For a HiRDB/Single Server, specifies in kilobytes the size of the shared memory area to be used by the single server for locking (lock pool).

For a HiRDB/Parallel Server, specifies in kilobytes the size of the shared memory area to be used for locking by back-end servers and dictionary servers. Use the `pd_fes_lck_pool_size` operand to specify the size of the shared memory area to be used for locking by front-end servers.

The area specified by this operand is used for tables for managing locked resources. Note the following relationship between the number of tables for managing locked resources, the number of lock requests, and the lock pool size:

Number of tables for managing locked resources = number of lock requests = size of lock pool × *coefficient*

Specification guidelines

- In the 32-bit mode, the amount of lock pool space required for six lock requests is 1 kilobyte.
- In the 64-bit mode, the amount of lock pool space required for four lock requests is 1 kilobyte.
- The following formulas can be used to determine the value to be specified for this operand:

HiRDB type	Formula
HiRDB/Single Server (32-bit mode)	↑↑ *a* ÷ *value of pd_lck_pool_partition* ↑ ÷ 6 ↑ × *value of pd_lck_pool_partition*
HiRDB/Parallel Server (32-bit mode)	↑↑ *b* ÷ *value of pd_lck_pool_partition* ↑ ÷ 6 ↑ × *value of pd_lck_pool_partition*
HiRDB/Single Server (64-bit mode)	↑↑ *a* ÷ *value of pd_lck_pool_partition* ↑ ÷ 4 ↑ × *value of pd_lck_pool_partition*
HiRDB/Parallel Server (64-bit mode)	↑↑ *b* ÷ *value of pd_lck_pool_partition* ↑ ÷ 4 ↑ × *value of pd_lck_pool_partition*

a: Total number of transaction lock requests to be executed concurrently by the single server. The number of lock requests depends on the SQL. For details about how to determine the total number of lock requests, see *D. Determining the Number of Locked Resources*.

b: Total number of transaction lock requests to be executed concurrently by each server (dictionary server or back-end server). The number of lock requests depends on the SQL. For details about how to determine the total number of lock requests, see *D. Determining the Number of Locked Resources*.

Note

When you execute `DROP TABLE` or `DROP SCHEMA` in a definition SQL, it is especially important to have already determined in advance an appropriate value for this operand.

Tuning the specified value

See the usage rate for the locked resources management table (`% OF USE LOCK TABLE`) displayed in the statistical information on system operation by the statistics analysis utility.# If the maximum usage rate equals or exceeds 80%, it is recommended that the operand's value be increased in preparation for future database expansion. If the maximum usage rate does not exceed 10%, it is recommended that the operand's value be decreased to conserve shared memory space.

#: For a HiRDB/Parallel Server, see the usage rate for the locked resources management table (`% OF USE LOCK TABLE`) displayed in the statistical information on system operation for each server for the dictionary servers and back-end servers.

Note

- If the value specified for this operand is too small, an SQL statement might return an error.
- Do not specify a larger value than necessary in this operand. A large value will increase the size of the shared memory used by HiRDB, which might cause a shortage of shared memory and prevent HiRDB from starting.
- If you do an all-item search of a table that contains many rows while locking is in units of rows, this operand's value will need to be increased to reflect the large number of items, which will increase the amount of memory required. Instead, consider making the following adjustments in the UAP:
  - Acquire locks in table units
  - If you can use the unlocked search facility, perform searches unlocked
  - Narrow the search conditions, and divide the processing into several transactions

Relationship to other operands

- When `v6compatible` is specified in the `pd_sysdef_default_option` operand, the default for this operand is `1024`.
- This operand is related to the `pd_lck_pool_partition` operand.

14) pd_fes_lck_pool_size = *front-end-server-lock-pool-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-2000000))**
- 64-bit mode: **((1-2000000000))**

This operand is applicable only to HiRDB/Parallel Server.

Specifies in kilobytes the size of the shared memory area to be used by front-end servers for locking (lock pool). Use the `pd_lck_pool_size` operand to specify the size of the shared memory area to be used for locking by single servers, dictionary servers, and back-end servers.

The area specified by this operand is used for tables for managing locked resources. Note the following relationship between the number of tables for managing locked resources, the number of lock requests, and the lock pool size:

Number of tables for managing locked resources = number of lock requests = size of lock pool × coefficient

Specification guidelines

- In the 32-bit mode, the amount of lock pool space required for six lock requests is 1 kilobyte.
- In the 64-bit mode, the amount of lock pool space required for four lock requests is 1 kilobyte.
- Use the following formula to determine the value of this operand:

  ↑↑ (*a* + *b*) ÷ *value of pd_fes_lck_pool_partition* ↑ ÷ *c* ↑ × *value of pd_fes_lck_pool_partition* (kilobytes)

  a: Total number of transaction lock requests to be executed concurrently by the front-end server. The number of lock requests depends on the SQL. For details about how to determine the total number of lock requests, see *D. Determining the Number of Locked Resources*.

  b: *(value of* `pd_max_users` + 3*)* × *(value of* `pd_max_access_tables` + 4*)*

  c: Use `6` and `4` for 32- and 64-bit modes, respectively.

Tuning the specified value

See the usage rate for the locked resources management table (`%OF USE LOCK TABLE`) displayed for the front-end server in the statistical information on system operation by the statistics analysis utility. If the maximum usage rate equals or exceeds 80%, it is recommended that the operand's value be increased in preparation for future database expansion. If the maximum usage rate does not exceed 10%, it is recommended that the operand's value be decreased to conserve shared memory space.

Operand default value

The default values of this operand are as follows:

- 32-bit mode

{(`pd_max_users` value + 3) × (`pd_max_access_tables` value + 4)} ÷ 6

- 64-bit mode

{(`pd_max_users` value + 3) × (`pd_max_access_tables` value + 4)} ÷ 4

Note

- If the value specified for this operand is too small, an SQL statement might return an error.
- Do not specify a larger value than necessary in this operand. A large value will increase the size of the shared memory used by HiRDB, which might cause a shortage of shared memory and prevent HiRDB from starting.

Relationship to other operands

This operand is related to the `pd_fes_lck_pool_partition` operand.

15) pd_lck_pool_partition = ***per-server-lock-pool-partition-count***

~<unsigned integer>((1-5000))<<1>>

Specify this operand to distribute lock processing.

For a HiRDB/Single Server, specifies the number of lock pool partitions to be used in locking by the single server when distributing lock processing.

For a HiRDB/Parallel Server, specifies the number of lock pool partitions to be used in locking by back-end servers and dictionary servers when distributing lock processing. Use the `pd_fes_lck_pool_partition` operand to specify the number of lock pool partitions to be used in locking by front-end servers.

For details about distributing lock processing, see the *HiRDB Version 9 System Operation Guide*.

Tuning the specified value

For details about how to tune the number of lock pool partitions, see the *HiRDB Version 9 System Operation Guide*.

Notes

- When the value set in this operand is too large, the size of the shared memory assigned to each lock pool partition becomes smaller, which might result in not enough lock pool partition capacity, causing SQL to return an error. Lock processing also takes time, so system performance declines. If this becomes an issue, specify a smaller value in this operand.
- The lock pool size must be at least 1 kilobyte. If a value larger than the value of `pd_lck_pool_size` is specified, the `KFPS00421-W` message will be issued and the value of `pd_lck_pool_size` will be assumed for this operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_pool_size`
- `pd_lck_deadlock_check_interval`

Effects on individual estimation formulas

If the value of the `pd_lck_pool_partition` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Processes started by HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

16) pd_fes_lck_pool_partition = ***front-end-server-lock-pool-partition-count***

~<unsigned integer>((1-5000))<<1>>

Specifies the number of lock pool partitions to be used in locking by front-end servers when distributing lock processing. Use the `pd_lck_pool_partition` operand to specify the number of lock pool partitions to be used in locking by single servers, dictionary servers, and back-end servers.

For details about distributing lock processing, see the *HiRDB Version 9 System Operation Guide*.

Tuning the specified value

For details about how to tune the number of lock pool partitions, see the *HiRDB Version 9 System Operation Guide*.

Notes

- When the value set in this operand is too large, the size of the shared memory assigned to each lock pool partition becomes smaller, which might result in not enough lock pool partition capacity, causing SQL to return an error. Lock processing also takes time, so system performance declines. If this becomes an issue, specify a smaller value in this operand.
- The lock pool size must be at least 1 kilobyte. If a value larger than the value of `pd_lck_pool_size` is specified, the `KFPS00421-W` message will be issued and the value of `pd_lck_pool_size` will be assumed for this operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_fes_lck_pool_size`
- `pd_lck_deadlock_check_interval`

Effects on individual estimation formulas

If the value of the `pd_fes_lck_pool_partition` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

17) pd_lck_until_disconnect_cnt = *total-number-of-tables-and-RDAREAs-to-be-locked-until-disconnect-specification*

~<unsigned integer>((0-140000))<<256>>

Specifies the number of resources to be locked for the tables and RDAREAs that are to be held across transactions. Based on the value specified for this operand, blocks for which lock with `UNTIL DISCONNECT` is specified for the tables and RDAREAs are allocated in the shared memory.

Specification guidelines

Normally, this operand need not be specified. Specification of a value other than the default value might be necessary in the following cases:

- When the number of utilities to be executed concurrently increases
- When a holdable cursor is used
- When the local buffer specified in the `pdlbuffer` operand is used
- When a shared RDAREA is used
- When an SQL session-specific temporary table is used

For details about how to estimate the specification value for this operand, see *C.5 Formula for determining total number of tables and RDAREAs per server locked with UNTIL DISCONNECT specified (pd_lck_until_disconnect_cnt)*.

Tuning the specified value

If the value specified for this operand is small, a transaction might roll back or a utility might terminate abnormally with return code `8`. In such cases, the message `KFPA11914-E` or `KFPH28001-E` is output. If this occurs, increase the value of this operand.

When the value of this operand is increased, the amount of required memory space increases proportionately. The required memory size can be expressed as follows: *value of this operand* × 48 (64 in the 64-bit mode) bytes.

Effects on individual estimation formulas

If the value of the `pd_lck_until_disconnect_cnt` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the number of records in a synchronization point dump file*
- *Formula 2* under *Formulas for shared memory used by a single server*
- *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 2* under *Formulas for the size of the shared memory used by a back-end server*

18) pd_max_open_holdable_cursors = *maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*

~<unsigned integer>((16-1024))<<16>>

When you use holdable cursors for a table for which a `LOCK` statement with the `UNTIL DISCONNECT` specification is not executed, this operand specifies the maximum number of holdable cursors that can be concurrently open for each transaction.

Note

Specifying a value other than the default value for this operand increases the amount of shared memory used.

Relationship to other operands

The values specified for this operand and the following operands are used for computing the shared memory size for lock servers. For a 32-bit mode HiRDB system, if the values specified for these operands are too large, the shared memory size of the lock servers exceeds 2 GB, and as a result, HiRDB might not start. Therefore, adjust the values specified for these operands so that the shared memory size of the lock servers does not exceed 2 GB.

- `pd_max_access_tables`
- `pd_max_users`
- `pd_max_bes_process`
- `pd_max_dic_process`
- `pd_lck_hash_entry`
- `pd_lck_pool_size`

For details about shared memory, see the *HiRDB Version 9 Installation and Design Guide*.

19) pd_lck_hash_entry = *lock-pool-hash-entry-count*

~<unsigned integer>((0-2147483647))<<0>>

Specifies the number of hash table entries to be used in the lock pool. According to the value specified here, HiRDB allocates a lock pool in the shared memory for the unit controller in each server (single server, front-end server, back-end server, and dictionary server).

Specification guidelines

Normally, omit this operand.

Consider specifying a value for this operand in the following cases:

- If you do not wish to change the shared memory size if possible when upgrading to version 06-02 or later, specify `11261`. In this case, the same number of hash entries is allocated as in the earlier version, the hash table size inside the lock pool remains the same as before.
- It is possible to improve performance by specifying in this operand a value greater than the recommended value shown below. However, specifying a value greater than variable *a* (also shown below) will not improve performance over the case in which *a* is specified.

  The recommended value is as follows:

  Recommended value = Largest prime number not exceeding MAX($\uparrow a \div 10 \uparrow$, 11,261)

Variable	Formula for computing the variable	
a	Single server	(`pd_max_users` value + 3) × (`pd_max_access_tables` value + 14) + `pd_lck_pool_size` × *c*

Variable	Formula for computing the variable		
	Front-end server	If `pd_fes_lck_pool_size` is omitted	(b + 3) × (`pd_max_access_tables` value + 4)
		If `pd_fes_lck_pool_size` is specified	`pd_fes_lck_pool_size` value × c
	Back-end server or dictionary server	(b + 3) × 10 + `pd_lck_pool_size` value × c	
b	Front-end server	Multiple front-end server	`pd_max_users` value + 1
		Not multiple front-end server	`pd_max_users` value
	Back-end server	If `pd_max_users` value > `pd_max_bes_process` value	`pd_max_users` value
		If `pd_max_users` value ≤ `pd_max_bes_process` value	`pd_max_bes_process` value
	Dictionary server	If `pd_max_users` value > `pd_max_dic_process` value	`pd_max_users` value
		If `pd_max_users` value ≤ `pd_max_dic_process` value	`pd_max_dic_process` value
c	6 for the 32-bit mode; 4 for the 64-bit mode		

Operand rules

- If this operand and the `pd_lck_hash_entry` operands of the server definitions are all omitted or `0` is specified in these operands, HiRDB calculates a recommended value for each server. (However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `11261`.)
- When a value that is neither `0` nor a prime number is specified in this operand, HiRDB assumes that the specification is the largest prime number that does not exceed the specified value.

Notes

If the value specified in this operand is too small, there might be an insufficient number of hash entries, and performance might deteriorate. If this operand is omitted, there will never be a shortage of hash entries and performance will not deteriorate due to an insufficient number of hash entries.

20) pd_dbsync_lck_release_count = *global-buffer-lock-release-interval-during-synchronization-point-processing*

~<unsigned integer>((0, 100-1073741824))<<10000>>

Specifies the interval for unlocking global buffers, when global buffer locking occurs during synchronization point processing.

During synchronization point processing, search processing occurs on the buffers (update buffers) and must be applied to the disk. Normally, global buffers are unlocked at a specific interval during search processing on the update buffers.

For example, if 100 is specified in this operand, a global buffer is unlocked once when search processing on 100 sectors (global buffer sectors) is completed. After that, the global buffer is locked again and search processing is resumed. In this example, unlocking occurs once every 100 sectors.

Advantage

By specifying this operand, you can adjust the global buffer lock time during synchronization point processing. When a small value is specified in this operand, the global buffer lock time becomes short and transaction performance might improve during synchronization point processing.

To obtain the global buffer pool lock time, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Buffer pool lock time during synchronization point processing (SYNCL).`

Specification guidelines

Normally, there is no need to specify this operand. Consider specifying this operand when both the following conditions apply:

- Transaction performance drops during synchronization point processing.
- A large number of buffer sectors is specified in the `-n` option of the `pdbuffer` operand.

Operand rules

- If the specified value is in the range 1 to 99, 100 is set automatically.
- If 0 is specified, global buffers are locked until update buffer search processing is completed.

Notes

If a small value is specified in this operand, the update buffer search time increases due to interrupts from other transactions and CPU usage rises. The global buffers updated during that time are also output during synchronization point processing. Therefore, the number of update buffers to be output during synchronization point processing increases. To obtain the number of update buffers to be output during synchronization point processing, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Number of synchronization point output pages (SYNCW)`. As you increase the value of this operand, the lock time for a global buffer to determine the buffer to be output at a synchronization point will increase. For this reason, lock contention grows substantially during synchronization point processing, which might affect transaction performance.

4.2.5 Operands related to buffers

21) pd_sql_object_cache_size = *SQL-object-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((22-256000))**
- 64-bit mode: **((22-2000000))**

Specifies in kilobytes the size of the buffer area (shared memory) in which SQL objects are to be placed.

Specification guidelines

- SQL objects are saved in a buffer until the user's transaction has terminated. The buffer must be large enough to store the SQL objects of all transactions that will be executed concurrently.
- When the hit rate is low for SQL object buffers, performance might degrade due to the overhead required for SQL parsing processing.
- SQL analysis can be reduced by saving the SQL objects of static SQLs in the buffer after transaction termination (until the buffer runs out of space) and sharing them among multiple users who execute the same UAP. To effectively utilize the buffer, allocate it so that the SQL objects of frequently-used UAPs are resident in the buffer.
- To estimate the buffer size, first determine the buffer size needed for UAP execution from the length of the SQL objects from the SQL statements to be issued by the UAP. Then, compute the buffer size by considering the number of UAPs that will be executed concurrently and the number of concurrently executing users.
- For details about how to estimate the length of the SQL object from a single SQL statement, see *C.2 Formulas for determining size of SQL object buffer (pd_sql_object_cache_size).*

Operand default

When this operand is omitted, the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the following value is assumed:

(*value of pd_max_users* + 3) × 22

Tuning the specified value

For details about tuning the SQL object buffer size, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_sql_object_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula 1* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 1* under *Formulas for the size of the shared memory used by a back-end server*

- *Formula for the shared memory used by a front-end server*

22) pd_table_def_cache_size = *table-definition-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((100-65535))**
- 64-bit mode: **((100-2000000))**
- HiRDB/Single Server

$$\langle\!\langle \uparrow \sqrt{\textit{value of pd\_max\_users}} \uparrow \times 100 + 3 \rangle\!\rangle$$

- HiRDB/Parallel Server

$$\langle\!\langle \uparrow \sqrt{(\textit{value of pd\_max\_bes\_process or pd\_max\_dic\_process, whichever is larger})} \uparrow \times 100 + 3 \rangle\!\rangle$$

Specifies in kilobytes the size of the buffer (shared memory) for table and sequence generator definition information. This information is used during SQL statement pre-processing. Definition information stored in this buffer is managed by the LRU method.

Advantages

- Table and sequence generator definition information that has been used is retained in memory as long as possible so that it can be used again without an input operation.
- Performance improves when a large number of dynamic SQLs are used.
- The number of communications with the dictionary server is reduced (in a HiRDB/Parallel Server).

Specification guidelines

- Specify the total size of the definition information needed for frequently-used tables and sequence generators.
- The table definition information buffer size per sequence generator is 8 kilobytes.
- For details about determining the size of the table definition information buffer per table, see *C.4 Formulas for determining size of table definition information buffer (pd_table_def_cache_size)*. For temporary tables, determine the size in the same manner as for non-partitioned tables.

Tuning the specified value

For details about how to tune the size of the table definition information buffer, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_table_def_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

23) pd_auth_cache_size = *user-privilege-information-buffer-size*

~<unsigned integer>((1-100))<<1>>(kilobytes)

Specifies in kilobytes the size of the buffer (shared memory) for user privilege information.

Specification guidelines

- The user privilege information buffer stores CONNECT privilege, DBA privilege, and audit privilege information. If this buffer contains no information, information is obtained from a dictionary table during HiRDB connection, thus lengthening the response time. Therefore, specify a buffer size that can store the information for the users who are always connected.
- Storing the user privilege information of each user requires 68 bytes. Use this information when computing the total buffer size.

Tuning the specified value

For details about how to tune the size of the user privilege information buffer, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_auth_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

24) pd_view_def_cache_size = *view-analysis-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((0-32000))**
- 64-bit mode: **((0-2000000))**

《$\uparrow \sqrt{(\textit{value of pd\_max\_users} + 3)} \uparrow \times 8$》

Specifies in kilobytes the size of the buffer (shared memory) for view analysis information.

Advantage

View analysis information that has been used is kept in the shared memory and can be used subsequently without an I/O operation.

Specification guideline

The total size of the view analysis information for frequently-used view tables is specified. For details about determining the size of the view analysis information buffer, see *C.3 Formulas for determining size of view analysis information buffers (pd_view_def_cache_size)*.

Tuning the specified value

For details about how to tune the size of the buffer for view analysis information, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_view_def_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

25) pd_type_def_cache_size = *user-defined-type-information-buffer-size*

~<unsigned integer><<0>> (kilobytes)

- 32-bit mode: **((100-65536))**
- 64-bit mode: **((100-2000000))**

This operand is applicable to user-defined types; specification of this operand is recommended when user-defined types will be used.

Specifies in kilobytes the size of the buffer (shared memory) for information on user-defined types. When a user-defined type is used, information on it is stored in this buffer. This information is used during pre-processing of SQL statements.

Advantages

- When user-defined type information is stored in this buffer, it is not necessary to access the dictionary table when the same user-defined type is used subsequently, thus reducing the number of input operations and the CPU usage time.
- The number of communications with the dictionary server is reduced.
- Specification of this operand improves performance when many dynamic SQLs are used.

Specification guidelines

Specify the total size of the definition information for frequently-used user-defined types that are defined in tables. The following formula can be used to determine the definition information size for one user-defined type:

$\uparrow \{((0.3 + 0.2 \times a + 0.1 \times b) + 3) \div 4\} \uparrow \times 4$ (kilobytes)

a: Number of user-defined type attributes *b*: Number of subtypes that have inherited a supertype

Tuning the specified value

For details about how to tune the size of the buffer for user-defined type information, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_type_def_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

26) pd_routine_def_cache_size = *routine-definition-information-buffer-size*

~<unsigned integer><<100>> (kilobytes)

- 32-bit mode: **((0, 20-65536))**
- 64-bit mode: **((0, 20-2000000))**

Specifies in kilobytes the size of the buffer (shared memory) for storing the following types of definition information (this information is used during pre-processing of SQL statements):

- Plug-in facility definition information
- System definition scalar facility definition information
- Routine definition information

Advantages

- When these types of definition information are stored in this buffer, it is not necessary to access the dictionary table when the same information is used subsequently, thus reducing the number of I/O operations and CPU usage time.
- The number of communications with the dictionary server is reduced.

Application criteria

Specify this operand when a large number of the following types of SQL statements are used:

- SQL statements that use a plug-in
- SQL statements that use the system definition scalar facility
- SQL statements that use a routine

Specification guidelines

For details about how to determine the value for this operand, see *C.6 Formulas for determining size of routine definition information buffer (pd_routine_def_cache_size)*.

Tuning the specified value

For details about how to tune the size of the routine definition information buffer, see the *HiRDB Version 9 System Operation Guide*.

Note

If the value specified for this operand is smaller than the total of the sizes of the definition information for all plug-ins, the definition information on plug-in facilities will not be allocated in the buffer.

Effects on individual estimation formulas

If the value of the `pd_routine_def_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

27) pd_registry_cache_size = *registry-information-buffer-size*

~<unsigned integer>((0-65536))<<10>> (kilobytes)

This operand is related to plug-ins. If you use a plug-in that uses registry information, we recommend that you use this operand. We also recommend that you use this operand if you use HiRDB Text Search Plug-in.

Specify the size of the buffer (shared memory) for storing registry information (in kilobytes). When registry information is used, it is stored in the buffer. Registry information is used during the execution of an SQL statement.

Advantages

- Once registry information is stored in this buffer, it is not necessary to access the registry when the same information is used subsequently, thus reducing the number of I/O operations and CPU usage time.
- The number of communications with the dictionary server is reduced.
- Specifying this operand can improve performance when a plug-in that makes frequent use of registry information is used.

Specification guidelines

Use the following formula to determine the registry information buffer size:

$\uparrow (0.3 + a) \uparrow \times b$ (kilobytes)

a: Average registry key length (kilobytes)

The average registry key length can be determined with the following SQL statement:

```
SELECT AVG(KEY_LENGTH) FROM MASTER.SQL_REGISTRY
```

Because the result of this SQL statement is output in bytes, convert it to kilobytes.

b: Number of registry keys registered

The number of registry keys registered can be determined with the following SQL:

```
SELECT COUNT(*)FROM MASTER.SQL_REGISTRY
```

Tuning the specified value

For details about how to tune the size of the buffer for registry information, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_registry_cache_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*
- *Formula for the shared memory used by a front-end server*

4.2.6 Operands related to shared memory

28) pd_sds_shmpool_size = *single-server-shared-memory-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-200000))**
- 64-bit mode: **((1-4000000000))**

This operand is applicable only to HiRDB/Single Server, and specifies the size of the area to be used (in kilobytes) by a single server as part of the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand. If it is omitted, HiRDB computes a value for it (however, if `v6compatible` is specified for the `pd_sysdef_default_option` operand, `1024` is assumed). HiRDB computes this value based on the values for the related definitions. Therefore, specify the appropriate values for the related definitions. For details about the related definitions, see *Formulas for shared memory used by a single server* in the *HiRDB Version 9 Installation and Design Guide*.

If the values of any of the operands of the variables used in the estimation formula are changed, HiRDB automatically re-calculates this operand's value.

Tuning the specified value

If any of the following messages is output, increase the specification value of this operand.

- `KFPA20003-E`
- `KFPD00005-E`

- `KFPD00012-E`
- `KFPD00021-E`
- `KFPH20003-E`

Notes

If this operand is omitted and the appropriate values are not specified in the related definitions, the value obtained by HiRDB might be too large, resulting in the allocation of an unnecessarily large amount of shared memory, or too small, resulting in one or more of the following problems:

- HiRDB does not start.
- A UAP or utility does not execute.

If the value specified for this operand is unnecessarily large or unnecessarily small, the same problems also occur.

Effects on individual estimation formulas

If the value of the `pd_sds_shmpool_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a single server*

29) pd_dic_shmpool_size = *dictionary-server-shared-memory-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-200000))**
- 64-bit mode: **((1-4000000000))**

This operand is applicable only to HiRDB/Parallel Server. It specifies the size of the area (in kilobytes) to be used by a dictionary server as part of the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand. If it is omitted, HiRDB computes a value for it (however, if `v6compatible` is specified for the `pd_sysdef_default_option` operand, `1024` is assumed). HiRDB computes this value based on the values for the related definitions. Therefore, specify the appropriate values for the related definitions. For details about the related definitions, see *Formulas for the size of the shared memory used by a dictionary server* in the *HiRDB Version 9 Installation and Design Guide*.

If the values of any of the operands of the variables used in the estimation formula are changed, HiRDB automatically re-calculates this operand's value.

Tuning the specified value

If any of the following messages is output, increase the specification value of this operand.

- `KFPA20003-E`
- `KFPD00005-E`
- `KFPD00012-E`
- `KFPD00021-E`
- `KFPH20003-E`

Notes

If this operand is omitted and the appropriate values are not specified in the related definitions, the value obtained by HiRDB might be too large, resulting in an allocation of unnecessarily large shared memory, or too small, resulting in the following problems:

- A unit does not start.
- A UAP or utility does not execute.

If the value specified for this operand is unnecessarily large or unnecessarily small, the same problems also occur.

30) pd_bes_shmpool_size = *back-end-server-shared-memory-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-200000))**

- 64-bit mode: **((1-4000000000))**

This operand is applicable only to HiRDB/Parallel Server. Specifies the size of the area (in kilobytes) to be used by a back-end server as part of the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand. If it is omitted, HiRDB computes a value for it (however, if `v6compatible` is specified for the `pd_sysdef_default_option` operand, `1024` is assumed). HiRDB computes this value based on the values for the related definitions. Therefore, specify the appropriate values for the related definitions. For details about the related definitions, see *Formulas for the size of the shared memory used by a back-end server* in the *HiRDB Version 9 Installation and Design Guide*.

If the values of any of the operands of the variables used in the estimation formula are changed, HiRDB automatically re-calculates this operand value.

Tuning the specified value

If any of the following messages is output, increase the specification value of this operand.

- `KFPA20003-E`
- `KFPD00005-E`
- `KFPD00012-E`
- `KFPD00021-E`
- `KFPH20003-E`

Notes

If this operand is omitted and the appropriate values are not specified in the related definitions, the value obtained by HiRDB might be too large, resulting in an allocation of unnecessarily large shared memory, or too small, resulting in the following problems:

- A unit does not start.
- A UAP or utility does not execute.

If the value specified for this operand is unnecessarily large or unnecessarily small, the same problems also occur.

4.2.7 Operands related to RPC trace information

31) pd_rpc_trace = Y | N

Specifies whether RPC trace information is to be collected. HiRDB maintenance information is output in the RPC trace information.

Normally, omit this operand.

`Y`: Collect RPC trace information.

`N`: Do not collect RPC trace information.

Note

Specifying `Y` for this operand degrades communication performance.

Operand default value

If this operand is omitted, the value specified for the same operand in the unit control information definition or the system common definition takes effect, in that order. If the same operand is also omitted from the unit control information definition and the system common definition, `N` is assumed.

32) pd_rpc_trace_name = "*name-of-RPC-trace-collection-file*"

~<path name of up to 254 characters>

Specifies as an `absolute` path name the file name for the RPC trace files. Three RPC trace files are created, with 1, 2, and *l* suffixed to the specified file name.

Note

Files with a maximum size of the *pd_rpc_trace_size value* × 2 are created under the directory specified by this operand. Pay attention to the file size.

Operand default value

If this operand is omitted, the value specified for the same operand in the unit control information definition or the system common definition takes effect, in that order. If the same operand is also omitted from the unit control information definition and the system common definition, the following value is assumed:

`%PDDIR%\spool\rpctr`

33) pd_rpc_trace_size = *RPC-trace-collection-file-size*

~<unsigned integer>((1024-2147483648)) (bytes)

Specifies in bytes the size of the RPC trace files.

The value specified by this operand determines the size of both RPC trace file 1 and RPC trace file 2.

Specification guidelines

An RPC trace collects in a single file in time order the information on the communications among all processes. If the volume of this information exceeds the specified value, trace data output is switched to the other file, which means that older trace information will be overwritten. When this happens, there might not be enough trace information available, making troubleshooting difficult. For this reason, specify at least `1000000` for this operand.

Note

The value specified by this operand does not apply to RPC trace file *l*, because its size is fixed at 0 bytes.

Operand default value

If this operand is omitted, the value specified for the same operand in the unit control information definition or the system common definition takes effect, in that order. If the same operand is also omitted from the unit control information definition and the system common definition, `4096` is assumed.

4.2.8 Operands related to troubleshooting information

34) pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*

~<unsigned integer>((126-16383))

A HiRDB process records the history of the executed functions and macros inside the process private memory. This history is called a *module trace*. This operand specifies the number of module trace records. The content of this history is loaded into the `core` file and is output when a process error occurs.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the unit control information definition or in the system common definition, in that order, is assumed. When the same operand is also omitted in the unit control information definition and the system common definition, the default is `126`.

Note

Process private memory of the following size is allocated to each process:

In the 32-bit mode: 64 + 48 × *pd_module_trace_max operand value* (bytes)

In the 64-bit mode: 64 + 64 × *pd_module_trace_max operand value* (bytes)

35) pd_module_trace_timer_level = 0 | 10 | 20

Specifies how to acquire the time to be output in module traces. The following table explains the meaning of the value specified for this operand.

Specified value	Time acquisition method
`0`	Time is output in seconds at every module trace output location.
`10`	Time is output in microseconds only at performance-critical module trace output locations, such as those before and after input/output processing, and time is output in seconds at other locations.
`20`	Time is output in microseconds at every module trace output location.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the unit control information definition or in the system common definition, in that order, is assumed. When the same operand is also omitted in the unit control information definition and the system common definition, the default is `0`.

Note

If you specify a value other than `0` for this operand, a function for acquiring time in microseconds is issued, and as a result, system performance might decline.

36) pd_pth_trace_max = *maximum-number-of-stored-communication-traces*

~<unsigned integer>((1024-8388608))

Specifies the maximum number of communication trace records to be used as troubleshooting information.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reason such as performance checking, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the unit control information definition or in the system common definition, in that order, is assumed. When the same operand is also omitted in the unit control information definition and the system common definition, the default is `1024`.

Notes

Increasing the value of this operand increases the amount of process private memory secured by HiRDB processes.

Process private memory for communication traces is calculated based on this operand's value rounded up to the power of two. For details about memory requirements, see *Calculation of required memory* in the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_pth_trace_max` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

4.2.9 Operands related to global buffers

37) pd_max_add_dbbuff_no = *maximum-global-buffers-count-for-dynamic-addition*

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, this operand specifies the maximum number of global buffers (per server) that can be added dynamically by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

- Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify a sufficient value based on that value.
- Determine the operand's value in such a manner that the following condition is satisfied:
  Value of `pd_max_add_dbbuff_no` $\leq$ 2,000,000 - number of global buffers allocated per server during HiRDB startup

Operand default value

The default value of this operand is as follows:

Condition		Default value
32-bit mode	*a* ≥ 500	`256`
	a < 500	500 - *a*
64-bit mode	*a* ≥ 1,000	`256`
	a < 1,000	1,000 - *a*

a: Number of global buffers allocated per server during HiRDB startup

Notes

Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the value of S* under *Determining the size of status files*
- *Formula 2*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for shared memory used by a single server*
- *Formula 2*, *Formula 3*, *Formula 4*, and *Formula 5* under *Formulas for the size of the shared memory used by a dictionary server*
- *Formula 2*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for the size of the shared memory used by a back-end server*

38) pd_max_add_dbbuff_shm_no = *maximum-shared-memory-segments-count-for-dynamic-addition*

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, specifies the maximum number of shared memory segments (per server) that can be allocated when dynamic addition is performed by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify an appropriate value.

Operand default value

The default value of this operand is as follows:

Condition		Default value
`pd_max_add_dbbuff_no` operand is omitted.	32-bit mode	500 + *A*
	64-bit mode	1,000 + *A*
`pd_max_add_dbbuff_no` operand is specified.	32-bit mode	↓ *value of pd_max_add_dbbuff_no* × 1.5 + *A* ↓ (If the value is 32,752 or greater, `32752` is set)
	64-bit mode	

A: Remaining number of shared memory segments that can be allocated during HiRDB startup. This value can be calculated using the following formula:

A = *a* - *b*

Assign the following values to *a* and *b*:

a = *value of pd_max_dbbuff_shm_no* (in 64-bit mode, 16)

b = *number of shared memory segments allocated to each server during HiRDB startup*

You can obtain information about the shared memory segments by using the `pdls -d mem` command or an OS command.

Notes

- If the following condition is satisfied, the value of the `pd_max_add_dbbuff_no` operand is assumed in this operand:

  Value of `pd_max_add_dbbuff_shm_no` < value of `pd_max_add_dbbuff_no`

  The value of the `pd_max_add_dbbuff_shm_no` operand is also assumed when the default value satisfies the above condition.
- Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.
- If the size of a shared memory segment to be added exceeds the `SHMMAX` operand value, shared memory is divided into multiple segments based on the `SHMMAX` operand value as the maximum value. Either increase the value of the `SHMMAX` operand based on the size of the shared memory segment to be added or increase the value of the `pd_max_add_dbbuff_no` operand so that no shortage occurs when the shared memory is segmented.
- If the facility for dynamically changing global buffers is used, the total number of shared memory segments that are allocated for the global buffers can be calculated using the following formula:

  pd_max_dbbuff_shm_no value + *pd_max_add_dbbuff_shm_no value*

  Therefore, if more shared memory segments were allocated when HiRDB started than the number specified for `pd_max_dbbuff_shm_no`, the number of shared memory segments that are actually allocated during dynamic change is the value of `pd_max_add_dbbuff_shm_no` minus the excess number of shared memory segments.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_no`
- `pd_max_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_shm_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

4.2.10 Operands related to temporary tables

39) pd_max_temporary_object_no = *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*

~<unsigned integer> ((0-131072))

Specifies the maximum number of temporary tables and temporary table indexes that can be used at any one time for each server.

Specification guidelines

Use the formula shown below to determine the value of this operand. For a HiRDB/Parallel Server, obtain the value for each back-end server.

Maximum value of (*the number of transaction-specific temporary tables*

+ *total number of temporary table indexes for those temporary tables*) $\times$ *number of activities*#

+ (*number of SQL session-specific temporary tables used in the SQL session*

+ *total number of temporary table indexes for those temporary tables*) $\times$ *number of connected users*

#

number of activities:

((*value of pd_max_users* + 3) $\times$ 2 + 1) + α

α : If the value specified for `pd_max_users` is `60` or less, `5`; if it is `61` or greater, `0`.

For a HiRDB/Parallel Server, this operand value is applied to each back-end server. Therefore, as a guideline, specify the largest value used among all back-end servers in this operand.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `0` is assumed.

Notes

The value of this operand affects the size of shared memory used by HiRDB; therefore, do not specify a value that is greater than the value described in *Specification guidelines*. If the specified value is greater than the value described in *Specification guidelines*, HiRDB might not be able to start due to a shortage of shared memory.

Relationship to other operands

This operand is related to the `pd_max_tmp_table_rdarea_no` operand.

Effects on individual estimation formulas

If the value of the `pd_max_temporary_object_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 8* under *Formulas for shared memory used by a single server*
- *Formula 8* under *Formulas for the size of the shared memory used by a back-end server*

4.2.11 Operands related to security

40) pd_audit_def_buffer_size = *security-audit-information-buffer-length*

~<unsigned integer>((1-2000000)) (kilobytes)

Specifies (in kilobytes) the buffer size (shared memory) used for storing information for the security audit facility.

Specification guidelines

Use the following formula to determine the security audit information buffer length:

$\uparrow 0.3 + a \times 0.25 \uparrow$ (kilobytes)

a: Number of objects for the narrowing condition that was specified in the audit trail of the security audit facility

Default value

If this operand is omitted, the buffer size shown below is acquired during HiRDB startup. If this amount of memory cannot be allocated, HiRDB will start, but the security audit information buffer is not created. In this case, the `KFPD00032-W` message is issued.

$\uparrow 0.3 + \max\{(a + 100), (a \times 1.2)\} \times 0.25 \uparrow$ (kilobytes)

a: Number of objects for the narrowing condition that was specified in the audit trail of the security audit facility

Notes

If the amount of memory required by this operand's specification cannot be allocated, HiRDB will not start.

Effects on individual estimation formulas

If the value of the `pd_audit_def_buffer_size` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

4.2.12 Operands related to Java

Java operands are specified when a Java stored procedure or Java stored function is used. For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide.*

41) pd_java_stdout_file = *"Java-virtual-machine-standard-output-and-standard-error-output-destination-file"* ~<path name>

Specifies as an absolute path name the file to which the standard output and standard error output are to be output in a Java virtual machine.

Specification guideline

Because the size of the file specified by this operand is extremely large, this operand is not normally specified. It is recommended that this operand be specified during debugging of a Java stored procedure or Java stored function. There is no limit to the size of the file that can be specified by this operand.

Operand default

When this operand is omitted, the specification of the same operand in the unit control information definition or in the system common definition, in that order, is assumed. When the same operand is also omitted in the unit control information definition and the system common definition, the Java Virtual Machine standard output and standard error output are ignored.

Note

If there are simultaneous writing attempts from multiple processes, their output contents cannot be guaranteed.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

42) pd_java_castoff = Y | N

Specifies whether to use the following events as triggers for shutting down the process at the server (single server, front-end server, dictionary server, or back-end server) that started the Java Virtual Machine:

No.	Server type	Process name	Trigger that ends process
1	Single server	`pdsds`	UAP is disconnected
2	Front-end server	`pdfes`	UAP is disconnected
3	Dictionary server	`pddic`	Transaction is completed or UAP is disconnected
4	Back-end server	`pdbes`	Transaction is completed or UAP is disconnected

`Y`: Shut down server process when trigger occurs.

`N`: Do not shut down server process when trigger occurs.

Specification guidelines

Normally, this operand is not specified. However, if you encounter the problems described below, we recommend that you specify `Y` in this operand:

- Use of the Java Virtual Machine causes the amount of memory usage to increase to the point where the available system memory becomes nearly exhausted.

- SQL code that includes numerous search conditions is executed, and even though the connection does not use the Java Virtual Machine, the maximum stack size set by the Java Virtual Machine on another connection prevents the stack from expanding, causing the server process to be aborted by a segmentation error.

For details about the Java Virtual Machine facility, see the Java Virtual Machine documentation.

Notes

On systems that run Java stored routines frequently, specifying Y in this operand will generate overhead for server process restarts and Java Virtual Machine startups.

Relationship to other operands

This operand is related to the `pd_process_count` operand.

4.2.13 Operands related to system log files

43) pd_log_dual = Y | N

Specifies whether dual system log files are to be used.

`Y`: Use dual system log files.

`N`: Do not use dual system log files.

Advantages

When dual system log files are used, HiRDB collects the same system log information in both files, which are called File A and File B. If a failure occurs in one of the files while the collected system log file is being loaded, the file can be loaded from the other file, resulting in higher system reliability.

Relationship to other operands

When use of dual system log files is specified, the name of the File B system log file must be specified with the `pdlogadpf` operand in each server definition.

44) pd_log_remain_space_check = warn | safe

Specifies the processing to be performed by HiRDB when the available space in the system log file falls below the warning level. This is called the facility for monitoring the free area for the system log file. For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

`warn`:

When the available space in the system log file falls below the warning level, the `KFPS01162-W` message is output.

`safe`:

When the available space in the system log file falls below the warning level, scheduling of new transactions is suppressed, all transactions inside the server are forcibly terminated, and the `KFPS01160-E` message is output.

Specification guideline

We recommend that you specify `safe` because it can reduce the probability of abnormal termination of units due to system log file space shortage. However, when `safe` is specified, all transactions inside the server are forcibly terminated when a shortage occurs in the available space in the system log file. Therefore, the design of the system log file requires more accuracy. For details about system log file design, see the *HiRDB Version 9 Installation and Design Guide*.

45) pd_log_auto_unload_restart = Y | N

Specifies whether the automatic log unloading facility is to be restarted when the system log files are swapped following the issuance of a message indicating that the automatic log unloading facility stopped due to an error (message `KFPS01150-E`).

`Y`: Restart the automatic log unloading facility.

`N`: Do not restart the automatic log unloading facility.

Condition

The following two conditions must be satisfied:

- `Y` is specified in the `pd_log_unload_check` operand or this operand is omitted.
- The `pd_log_auto_unload_path` operand is specified.

Advantages

- If `Y` is specified in this operand and a shortage of disk capacity occurs due to an increase in the number of work files, or because the unload processing fails due to a temporary error such as a process creation error, HiRDB automatically restarts the unloading of system logs the next time the system log files are swapped.
- If `N` is specified in this operand and the unload log files are set to be saved while the automatic log unloading facility is stopped due to an error, you can prevent the unload log files from being saved before the `pdlogatul -t` command is executed.

Specification guidelines

Normally, specify `Y` or omit this operand.

Specify `N` if you have already set HiRDB to monitor the automatic log unloading facility termination message (`KFPS01150-E` message) and restart the facility with the `pdlogatul -b` command.

46) pd_log_singleoperation = Y | <u>N</u>

This operand is applicable when dual system log files are used; it need not be specified when dual system log files are not used.

Specifies whether the single-operation mode is to be used for the system log files. Even if an error occurs in a system log file, making no dual system log files available, HiRDB (or a unit for a HiRDB/Parallel Server) can continue processing using the remaining single normal system log file without being abnormally terminated. This is called *single operation of the system log files*.

The mode in which continuation of processing is permitted only with both system log files available (normal operation mode) is called *double operation of the system log files*.

`Y`: Use single operation of the system log files.

`N`: Do not use single operation of the system log files. Both system log files must always be used.

Condition

`Y` must be specified in the `pd_log_dual` operand.

47) pd_log_rerun_reserved_file_open = Y | <u>N</u>

Specifies whether a system log file is to be opened automatically.

If no system log file that can be overwritten is available during a HiRDB (or unit) restart, HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of system log file*.

A reserved file is used in the following cases:

- Between the time of a restart and the time when the first synchronization point dump is collected
- When none of the opened file groups can be overwritten

`Y`: Open a system log file automatically (open and use a reserved file).

`N`: Do not open a system log file automatically (do not use a reserved file).

Advantages

When `Y` is specified, the unit can be restarted as long as a reserved file is available, even though no file that can be swapped in is available during the unit restart.

However, if a file that is in unload wait status is available, the unit is stopped; once that file has been unloaded, the unit can be restarted.

48) pd_log_rerun_swap = Y | <u>N</u>

Specifies whether the system log files are to be swapped during a HiRDB (or unit) restart.

`Y`: Swap the system log files.

`N`: Do not swap the system log files.

Advantage

When `Y` is specified, there can be a physical separation between the system log files used before and after a restart. Therefore, the system log file that was being used before the restart can be reused during server operation.

49) pd_log_swap_timeout = *wait-time-for-completion-of-system-log-file-swapping*

~<unsigned integer>((1-32580))<<180>> (seconds)

Specifies in seconds the wait time during which swapping of system log files must be completed. If a swap of system log files is not completed within the specified amount of time, the unit terminates abnormally.

Specification guidelines

Normally, there is no need to specify this operand. If it takes a long time to swap system log files for a reason such as poor machine performance, you can specify this operand with a value that is greater than the default value. To detect errors or delays quickly during system log file swapping so that the unit can be terminated (such as because of a disk failure), reduce the value of this operand.

50) pd_log_unload_check = Y | N

Specifies whether HiRDB is to check the unload status of system log files.

`Y`:

Check the unload status (normal operation).

`N`:

Do not check the unload status. A system log file is placed in swappable status, regardless of its unload status, when both of the following conditions are satisfied:

- It is in overwritable status
- It is in extraction completed status (HiRDB Datareplicator)

In such a case, the system log file operation method is to release checking of unload status. For details about this operation method, see the *HiRDB Version 9 System Operation Guide*.

Advantages

Specifying `N` provides the following advantages:

- Operations are simplified because the system log file unload operation is eliminated.
- It is not necessary to provide files for storing unload files.

Specification guideline

Specify `N` if the system log file will not be needed for database recovery (in other words, if recovery from a backup collection point will be sufficient).

Notes

The following points apply when `N` is specified:

- Database can be recovered only if backups have been made.
- If this option is specified when the system log file is required for database recovery, it will not be possible to recover the database.

51) pd_log_max_data_size = *log-input/output-buffer-size*

~<unsigned integer>((32000-523000))<<400000>> (bytes)

Specifies in bytes the size of the buffer to be used for system log input/output operations.

Specification guidelines

- HiRDB/Single Server
  Change the specification value according to the tuning method.
- HiRDB/Parallel Server
  Specify a value that satisfies conditional expression 1 below. If `uap` is specified in the `pd_rpl_reflect_mode` operand or a recovery-unnecessary front-end server is used, specify a value that satisfies both the conditional expressions below (1 and 2). To optimize the value, use the tuning method for the specification value.
  Conditional expression 1: log input/output buffer length $\geq$ *a*
  a: 72 $\times$ (*maximum number of back-end and dictionary servers that are targets for reference or update processing by a single transaction*) + 1,344
  Conditional expression 2: log input/output buffer length $\geq$ *b*
  b: (*maximum number of back-end and dictionary servers that are targets for reference or* update processing by a single transaction + 1) $\times$ 128 + 64

Tuning the specified value

A value (other than the default value) might need to be specified for this operand after the following types of statistics analysis utility statistical information related to system operation have been checked:

- Number of buffer sectors waiting for input/output (`# OF BUFFER FOR WAIT I/O`)
- Number of waits caused by lack of a current buffer (`# OF WAIT THREAD`)

If the number of waits caused by lack of a current buffer is not `0`, increase the value of this operand.

Notes

The specification of this operand affects the response and throughput of SQL code executed by a transaction. When a small value is specified, writing to system log files occurs frequently, which might result in deterioration of performance.

Relationship to other operands

- Use this operand and the `pd_log_write_buff_count` operand to determine the log I/O buffer size.
- If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `32000`.

52) pd_log_write_buff_count = *log-output-buffer-sectors-count*

~<unsigned integer>((3-65000))<<10>>

Specifies the number of buffer sectors to be used for system log output.

Tuning the specified value

Specify `10` (the default) for this operand initially. Thereafter, use a statistics analysis utility to obtain statistical information related to system operation to check the number of waits caused by a shortage of current buffer space (`# OF WAIT THREAD`). If the number of waits is high, specify a larger value to improve throughput.

Notes

If a small value is specified in this operand and the number of transactions is high, multiple transactions might be waiting for system log output, lowering system performance.

Relationship to other operands

- Use this operand and the `pd_log_max_data_size` operand to determine the number of log output buffer sectors.
- If `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, the default value for this operand is `3`.

53) pd_log_rec_leng = *system-log-file-record-length*

~<unsigned integer>((1024, 2048, 4096))<<4096>> (bytes)

Specifies the record length for the system log files; the specifiable values are `1024`, `2048`, and `4096`.

Specify the record length specified in the `-l` option of the `pdloginit` command for this operand.

Specification guidelines

Specify the record length based on the guidelines for designing the record length of system log files. For details about the guidelines for designing the record length of system log files, see *Record length of a system log file* in the *HiRDB Version 9 Installation and Design Guide*.

Notes

- If a value that is different from the record length specified by the `-l` option of the `pdloginit` command is specified for this operand, system log files cannot be opened.
- For details about how to modify the system log file record length, see the *HiRDB Version 9 System Operation Guide*.

54) pd_log_rollback_buff_count = *rollback-log-input-buffer-sector-count*

~<unsigned integer>((0-256))

Specifies the number of buffer sectors to be used for system log input during rollback processing. When `0` is specified in this operand, HiRDB determines the number of rollback log input buffer sectors.

Specification guidelines

- We recommend that you specify `0` in this operand, in which case HiRDB will calculate an appropriate value automatically.
- If the specified value is too small, concurrent execution of rollbacks might be slowed. If the specified value is too large, the unit controller might use more shared memory than is necessary.

Tuning the specified value

Specify `0` in this operand. If specifying `0` leads to memory shortages, do not specify this operand.

Operand defaults

Defaults for this operand are as follows:

- Standby-less system switchover (1:1) facility is used in the unit
  Number of alternate back-end servers × 2
- Standby-less system switchover (effects distributed) facility is used in the unit
  (*Number of host back-end servers* + *value of pd_ha_max_act_guest_servers*) × 2
- All other cases
  Number of servers in the unit × 2

Effects on individual estimation formulas

If the value of the `pd_log_rollback_buff_count` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

55) pd_log_auto_expand_size = *extension-amount-per-system-log-file-extension-trigger[,extension-limit]*

~<unsigned integer>((0-104857600))<<0,0>>(records)

Specify this operand when the system log file automatic extension facility is used.

Specifies the number of records to be added to the system log file for each extension trigger and an upper limit for extending file size.

If the amount to be added per trigger is omitted or `0` is specified, the system log file will not be extended automatically. If the extension limit is omitted or `0` is specified, the system log file might be extended until either the disk on which the file system area is located becomes full or the system log file capacity reaches its upper limit. When a value is specified for the amount to be added per trigger that is larger than the extension limit, the file will be extended up to the extension limit.

For details about the system log file automatic extension facility, see the *HiRDB Version 9 System Operation Guide*.

Condition

`Y` must be specified in the `pd_large_file_use` operand or specification of this operand must be omitted.

Specification guidelines

- Specify in this operand the amount to be added per trigger based on the number of records specified when the system log file was created with the `-n` option of the `pdloginit` command. Determine 10 percent of the average number of records for the system log file for each server and then specify the value for the server that has the largest value.

  ● In the case of a BES that uses the standby-less system switchover (1:1) facility, determine 10 percent of the average value including the system log file of the alternate BES.

  ● In the case of a BES that uses the standby-less system switchover (effects distributed) facility, determine 10 percent of the average value including the system log file of the guest BES.
- Normally, an extension limit is omitted.

Tuning the specified value

If the system log output volume exceeds the expanded size after automatic extension, the system log file could become full, resulting in a unitdown. In such a case, increase the specified value (number of records to be added per trigger). If extension processing requires so much time that it affects transaction performance, specify a smaller value.

4.2.14 Operands related to synchronization point dump files

56) pd_spd_dual = Y | <u>N</u>

Specifies whether to use dual synchronization point dump files.

`Y`: Uses dual synchronization point dump files.

`N`: Does not use dual synchronization point dump files.

Advantage

When dual synchronization point dump files are used, HiRDB collects the same synchronization point dump in both dump files A and B. Even when an error occurs in one of the files when the collected synchronization point dump is being loaded, the synchronization point dump can be loaded from the other file, thus improving system reliability.

Relationship to other operands

To use dual synchronization point dump files, specify the name of synchronization point dump file B in the `pdlogadpf` operand of each server definition.

57) pd_spd_assurance_msg = <u>Y</u> | N

Specifies whether the `KFPS02183-I` message is to be output when a synchronization point dump is completed.

`Y`: Output the message.

`N`: Do not output the message.

58) pd_spd_assurance_count = *number-of-guaranteed-valid-generations*

~<unsigned integer>((1-2))<<1>>

Specifies the range of system log files to be saved during HiRDB operation as a protection against events such as a synchronization point dump file input error during system recovery. What is specified here is a number of synchronization point dump file generations; the specified value is referred to as the *number of guaranteed-valid generations*. The number of past generations of synchronization point dump files specified here are overwrite disabled.

Advantage

When `2` is specified as number of guaranteed-valid generations and an error occurs in the most recent synchronization point dump file generation, the system can be recovered using the preceding synchronization point dump file generation, resulting in higher reliability.

Specification guidelines

- To improve reliability, specify `2` for the number of guaranteed valid generations. However, the number of overwrite disabled synchronization point dump files increases (to two).
- If reliability has been improved with the use of dual synchronization point dump files, we recommend that you omit this operand or specify `1` for it.

Notes

- The minimum number of synchronization point dump files needed is *number-of-guaranteed-valid-generations* + 1.
- Specifying `2` increases the number of overwrite disabled synchronization point dump files. Moreover, the system log files that correspond to the overwrite disabled synchronization point dump files are also overwrite disabled. Consequently, specifying `2` for the number of guaranteed valid generations increases the number of overwrite disabled system log files. As a result, a shortage might occur in the number of system log files that can be swapped in. To prevent this, it might be necessary to re-evaluate the system log file capacity.

59) pd_spd_reduced_mode = *reduced-mode-operation-option*

~<unsigned integer>((0-2))<<0>>

Specifies whether the reduced mode operation for synchronization point dump files is to be used.

Reduced mode operation is a facility for continuing processing if at least two synchronization point dump files can be used, even if an event such as a file error during HiRDB operation or restart reduces the number of synchronization point dump files to or below the number of required files (*number of guaranteed valid generations*# + 1).

#: Value specified for the `pd_spd_assurance_count` operand.

`0`: Do not use the reduced mode operation.

`1`: Use the reduced mode operation.

`2`: Use the reduced mode operation and issue a warning message whenever a synchronization point dump is being acquired during the reduced mode operation.

60) pd_spd_reserved_file_auto_open = Y | N

Specifies whether a synchronization point dump file is to be opened automatically. When `Y` is specified and an error in a synchronization point dump file reduces the number of synchronization point dump files to the number of files necessary for operation (*number of guaranteed valid generations*# + 1), HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of synchronization point dump file*.

#: Value specified for the `pd_spd_assurance_count` operand.

`Y`:

Open a synchronization point dump file automatically. A reserved file is opened when the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

`N`:

Do not open a synchronization point dump file. No reserved file is opened, even though the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

Relationship to other operands

This operand has a higher priority than the `pd_spd_reduced_mode` operand.

61) pd_spd_max_data_size = *synchronization-point-dump-file-buffer-size*

~<unsigned integer>((32000-4000000))<<32768>> (bytes)

Specifies in bytes the size of the buffer (shared memory) to be used for synchronization point dump file input/output operations.

The value specified here affects the number of synchronization point dump file input/output operations.

Specification guidelines

- Normally, this operand need not be specified.
- When the value of this operand is increased, the number of synchronization point dump file input/output operations is reduced.

62) pd_log_sdinterval = *system-log-output-volume*[,interval]

Specifies the collection interval for synchronization point dumps. This operand can be specified based on the following information:

- Volume of system log information
- Amount of time that has elapsed since the previous synchronization point dump was collected

***system-log-output-volume*: ~<unsigned integer>((100-100000))<<5000>> (number of log blocks)**

Specifies an interval between synchronization point dumps in terms of the number of blocks of log information. A synchronization point dump is collected each time system log information equivalent to the number of log blocks specified here has been output.

***interval*: ~<unsigned integer>((0 or 10-1440))<<60>> (minutes)**

Specifies a synchronization point dump collection interval in terms of the number of minutes between synchronization point dumps. A synchronization point dump is collected when the time interval specified here has elapsed since the previous synchronization point dump was collected.

- If `0` is specified for the interval, HiRDB does not use a time interval for collecting synchronization point dumps.
- If no transactions execute during the time interval since the previous synchronization point dump was collected, no synchronization point dump is collected, even if the interval specified here has elapsed.

Specification guidelines

- This operand need not be specified if no specific amount of time is specified for restarting HiRDB.
- The value specified for this operand affects the amount of time required to restart HiRDB.

  Specifying a small value for this operand reduces the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps increases, online performance might deteriorate in some cases.

  Conversely, specifying a large value for this operand increases the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps decreases, online performance might improve in some cases.

Tuning the specified value

The synchronization point dump collection intervals can be checked with the statistics analysis utility; the relevant information is shown under `SYNC POINT GET INTERVAL` in the statistical information related to system operation. Use the average of the `SYNC POINT GET INTERVAL` values. If the synchronization point dump collection interval is determined to be too long, decrease the specification value; conversely, if it is determined to be too short, increase the specification value.

Note

- The synchronization point dump collection interval is determined based on the volume of system log information that is output. Therefore, committing data from memory to a database takes a long time during intervals that have few updating transactions. If an error occurs at that time, it will take longer to recover the transactions that were generated during that period. If this is a possibility, also use the `interval` value to set the synchronization point dump collection interval.
- When synchronization point dumps are collected, update pages are output from the global buffer, proportionately increasing the loads on the CPU, input/output processing, and lock time. For this reason, reducing the synchronization point dump collection interval might cause processing delays.

Relationship to other operands

If `v6compatible or v7compatible` is specified in the `pd_sysdef_default_option` operand, the default system log output volume becomes `10000`.

4.2.15 Operands related to server status files (when an error occurs)

For details about the measures to be taken when an error occurs in a status file, see the *HiRDB Version 9 System Operation Guide*.

63) pd_sts_initial_error = <u>stop</u> | continue | excontinue

When a server (single server, front-end server, dictionary server, or back-end server) starts, HiRDB performs a process of identifying the current server status file. This operand specifies the action that HiRDB takes when any of the following errors is detected during this identification process.

- No real server status file is found.
- An error is detected in the server status file.

The current file identification process is applied to the server status files specified by the `pd_sts_file_name_1-7` operands.

`stop`:

When an error is detected in the server status file during the current file identification process, the startup of the server is stopped and HiRDB (or a unit for a HiRDB/Parallel Server) is not started. In this case, first take a corrective action for the status file in which the error was detected, and then start HiRDB.

`continue` or `excontinue`:

Even when an error is detected in the server status file during the current file identification process, the startup of the server is continued if the current file is normal. However, the startup might be stopped depending on the value specified for the `pd_sts_singleoperation` operand (whether operation continues with a single status file). The following table shows the relationship to the `pd_sts_singleoperation` operand.

● **Relationship to the pd_sts_singleoperation operand**

pd_sts_singleoperation operand value	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
`continue`	When an error is detected in the server status file, HiRDB cannot identify the current file, and thus the startup of the server is stopped, and HiRDB (or a unit for a HiRDB/Parallel Server) is not started.	The HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.
`stop` (default value)	When an error is detected in the server status file, HiRDB identifies the current file and the startup of the server is continued. However, if File A or B satisfies	If HiRDB cannot identify the current file, the HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and

pd_sts_singleoperation operand value	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
	any of the conditions listed in the following table (cases in which HiRDB cannot identify the current file), the startup of the server is stopped, and HiRDB (or a unit for a HiRDB/Parallel Server) is not started.	`pd_sts_last_active_side` operands. Afterwards, start HiRDB.

- **When HiRDB cannot identify the current file**

pd_sts_initial_error operand value	File A status	File B status
`continue`	Error shutdown	Error shutdown
	Error shutdown	Open (initial state)
	Error shutdown	No real file
	Open (initial state)	Error shutdown
	Open (initial state)	No real file
	No real file	Open (initial state)
	No real file	Error shutdown
	No real file	No real file
`excontinue`	Error shutdown	Error shutdown
	Error shutdown	No real file
	No real file	Error shutdown
	No real file	No real file

Therefore, to minimize actions by the HiRDB administrator (to reduce the number of cases in which both the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands must be specified), specify the following:

- `pd_sts_initial_error = excontinue`
- `pd_sts_singleoperation = stop`

Specification guidelines

The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_initial_error operand value	
	stop	continue or excontinue
Processing by HiRDB during server startup	When an error is detected in a server status file, the startup of the server is stopped and HiRDB (or a unit for a HiRDB/Parallel Server) is not started.	Even when an error is detected in a server status file, the startup of the server is continued if the current file is normal.
Specification guideline	To improve system reliability, specify `stop`.	To simplify the error-handling actions during HiRDB startup, specify `continue` or `excontinue`.
Advantage	Guarantees that all server status files of the server are normal when the server starts. Therefore, if an error occurs in the current file after HiRDB has started, it can be swapped to a spare file.	Even when an error is detected in some server status file during server startup, HiRDB can start with the remaining normal files only. Therefore, HiRDB stop time can be shortened. In this case, because the number of spare files has become small, it is necessary to immediately repair the status files containing errors.
Disadvantage	Possibility increases that an error in a server status file stops the startup of HiRDB.	Because HiRDB might be running with only a small number of spare files, system reliability is low.

Item	pd_sts_initial_error operand value	
	stop	continue or excontinue
		Depending on the number of spare files available, it might not be possible to swap server status files.

Notes

- If both current files are abnormal, the startup of the server is stopped regardless of the value specified for this operand, and HiRDB (or a unit for a HiRDB/Parallel Server) is not started.
- Before starting HiRDB, do not initialize the current file with the `pdstsinit` command. If the current file is initialized, HiRDB cannot be restarted.
- If `excontinue` is specified for this operand and all generations of the status files are in single-operation mode and open (initial status), HiRDB does not start.

Remarks

The following figure shows the values specified for this operand, the processing performed by HiRDB, and the actions to be taken by the HiRDB administrator.

● Values specified for this operand and the processing performed by HiRDB

pd_sts_initial_error operand value	Status file error	pd_sts_singleoperation operand value	Can HiRDB identify the current file?	Specification of pd_sts_last_active_file	Does the pd_sts_last_active_file operand value match the latest file that can be opened?	Current file error	Specification of the pd_sts_last_active_side operand	Is the file specified for the pd_sts_last_active_side operand usable?	Numbers listed in the Table Processing performed by HiRDB and actions to be taken by HiRDB the administrator
`stop` (default value)	None	—	—	—	—	—	—	—	[1]
	Yes	—	—	—	—	—	—	—	[5]
`continue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
`excontinue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]

Legend:

—: Not applicable (the condition does not affect the processing performed by HiRDB)

- Processing performed by HiRDB and actions to be taken by the HiRDB administrator

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[1]	HiRDB startup processing continues.	None
[2]	HiRDB identifies the latest current file and continues the startup processing.	Make the file that is in the error-shutdown state into a spare file.
[3]	Using the file specified in the `pd_sts_last_active_file` operand as the current status file, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[4]	Using the files specified in the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands as the current status files, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[5]	Because `stop` is specified for the `pd_sts_initial_error` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000010` in message `KFPS01005-E`.
[6]	Because the current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000016` in message `KFPS01005-E`.
[7]	Because the normal current file identified by HiRDB does not match the file specified in the `pd_sts_last_active_side` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000017` in message `KFPS01005-E`.
[8]	Because the current file name identified by HiRDB does not match the file name specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000015` in message `KFPS01005-E`.
[9]	Because the normal current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000018` in message `KFPS01005-E`.

64) pd_sts_singleoperation = stop | continue

Specifies whether processing of server status files continues in the single operation mode. The status file single-operation mode means that processing is to continue using only the normal file (single file) when an error occurs in the status file and a spare file is not available. For details about the status file single-operation mode, see the *HiRDB Version 9 Installation and Design Guide*.

If an error occurs in one of the current files and a spare file is available, the status file is swapped and processing continues, regardless of the value specified for this operand (operation in the single-operation mode does not occur).

`stop`:

Does not permit operation in the single-operation mode. If operation in the single-operation mode is necessary, HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated. If HiRDB is abnormally terminated, allocate spare files and then restart HiRDB.

`continue`:

Enables operation in the single-operation mode. When the single-operation mode goes into effect, the message `KFPS01044-I` is output. If an error occurs in the normal file during operation in the single-operation mode, or if HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated while the status file is being updated, HiRDB cannot be restarted. Therefore, when the single-operation mode goes into effect, immediately allocate spare files.

Specification guidelines

- We recommend that you specify `stop` to increase system reliability. We also recommend that you increase the number of spare files to guard against errors in the current files.
- The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_singleoperation operand value	
	stop	continue
Specification guideline	To improve system reliability, specify `stop`.	Specify `continue` if it is important not to stop HiRDB.
Advantage	When an error occurs in one of the current files and a spare file is not available, operation in the single-operation mode does not occur and HiRDB is abnormally terminated. Consequently, the possibility of losing the content of the current files is reduced.	Even when an error occurs in one of the current files and a spare file is not available, processing can be continued. Therefore, the possibility that an error in the status file stops HiRDB is reduced.
Disadvantage	Possibility that an error in the status file stops HiRDB increases. However, increasing the number of spare files can reduce this possibility.	If an error occurs in the normal status file during operation in the single-operation mode or if HiRDB is abnormally terminated during updating of the status file, the content of the current file is lost and HiRDB cannot be restarted.

Relationship to other operands

The combination of the values specified for the `pd_sts_singleoperation` and `pd_sts_initial_error` operands determines the processing by HiRDB when an error occurs in the status file. Therefore, the values to be specified for these operands must be determined together.

4.2.16 Operands related to the BES connection holding facility

For details about the BES connection holding facility, see the *HiRDB Version 9 System Operation Guide*.

65) pd_bes_connection_hold = Y | <u>N</u>

This operand is applicable only to HiRDB/Parallel Server.

Specifies whether to use the BES connection holding facility.

`Y`: The BES connection holding facility is used. This setting can reduce the overhead of connection processing because the connection between the front-end and back-end servers is retained beyond synchronization points. Note that the number of memory segments, ports, and sockets that are also retained is the same as the number of connections that are retained.

`N`: The BES connection holding facility is not used. The connection between front-end and back-end servers is processed the first time an SQL statement is executed and the connection is released when a synchronization point is reached.

Specification guidelines

For the specification guidelines, see *Application criteria* under *BES connection holding facility (HiRDB/ Parallel Server only)* in the *HiRDB Version 9 System Operation Guide*.

Relationship to the client environment definition

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDBESCONHOLD` operand in the client environment definition. For details about the `PDBESCONHOLD` operand, see the *HiRDB Version 9 UAP Development Guide*.

Note

When you use the BES connection holding facility, make sure that the following condition is satisfied:

number-of-processes-in-each-back-end-server (value of the `pd_max_bes_process` operand) ≥ *number-of-all-front-end-server-processes* (value of the `pd_max_users` operand × *number-of-front-end-servers*)

If this condition is not satisfied, a shortage in the number of back-end server processes might cause an SQL error. Furthermore, if you plan to execute a program such as a utility while HiRDB is running, make sure that you also include the number of back-end server processes required by the utility.

66) pd_bes_conn_hold_trn_interval = ***back-end-server-connection-hold-time***

~<unsigned integer>((0-3600))<<1>>(seconds)

Specifies the BES connection holding period in seconds.

When the BES connection holding facility is used, HiRDB monitors the period between the termination of a transaction and the execution of the next transaction. If this period is within the specified value, the connection between the front-end server and the back-end server is maintained. However, if this period exceeds the specified value, the connection between the front-end server and the back-end server is terminated after the transaction is terminated.

If `0` is specified for this operand, the period is not monitored. The connection between the front-end server and the back-end server is terminated only when the connection between the front-end server and a client is terminated by `SQL DISCONNECT` (`xa_close` if the XA library is being used) or because the value of the `PDCWAITTIME` operand is exceeded.

5 Single Server Definition

This chapter explains the operands of the single server definition.

5.1 Operand formats

A single server definition defines information for a single server. This section explains the formats used to specify the operands of a single server definition. Note that the numbers in the following table correspond to the numbers assigned to the operands explained in *5.2 Operand explanations*.

For users who are creating HiRDB system definitions for the first time

The first step is to determine the values to be specified for the operands shown in boldface type. In principle, HiRDB can be started once the boldface operands have been specified.

No.	Format	Operand category
1	`[set pd_process_count =` *resident-processes-count*`[,` *resident-processes-count-at-server-startup*`]]`#	Processes
2	`[set pd_server_cleanup_interval =` *interval-for-stopping-nonresident-server-processes*`]`#	
3	`[set pd_max_ard_process =` *asynchronous-READ-process-count*`]`#	
4	`[set pd_dfw_awt_process =`*number-of-parallel-writes-for-deferred-write-processing*`]`	
5	`[set pd_work_buff_mode = each |pool]`#	Work tables
6	`[set pd_work_buff_size =` *work-table-buffer-size*`]`#	
7	`[set pd_work_buff_expand_limit =` *work-table-buffer-expansion-limit*`]`#	
8	`[set pd_watch_pc_client_time =` *maximum-client-request-wait-time*`]`#	System monitoring
9	`[set pd_spd_syncpoint_skip_limit =` *maximum-number-of-skipped-synchronization-point-dumps*`]`#	
10	`[set pd_dfw_syncpoint_skip_limit =` *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*#	
11	`[set pd_cwaittime_wrn_pnt =` *output-condition-for-SQL-runtime-warning-information (% specification)* `|` *output-condition-for-SQL-runtime-warning-information (time specification)*`]`#	SQL runtime warning output facility
12	`[set pd_uap_exerror_log_use = YES	NO]`#	Facility for output of extended SQL error information
13	`[set pd_lck_pool_size =` *server-lock-pool-size*`]`#	Lock
14	`[set pd_lck_pool_partition =` *per-server-lock-pool-partition-count*`]`#	
15	`[set pd_lck_until_disconnect_cnt =` *total-number-of-tables-and-RDAREAs-to-be-locked-per-server-UNTIL-DISCONNECT-specification*`]`#	
16	`[set pd_max_open_holdable_cursors =`*maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*`]`#	
17	`[set pd_lck_hash_entry =` *lock-pool-hash-entry-count*`]`#	
18	`[set pd_dbsync_lck_release_count =` *global-buffer-lock-release-interval-during-synchronization-point-processing*`]`#	
19	`[set pd_sql_object_cache_size =` *SQL-object-buffer-size*`]`#	Buffers
20	`[set pd_table_def_cache_size =` *table-definition-information-buffer-size*`]`#	

No.	Format	Operand category
21	`[set pd_auth_cache_size =` *user-privilege-information-buffer-size*`]`#	
22	`[set pd_view_def_cache_size =` *view-analysis-information-buffer-size*`]`#	
23	`[set pd_type_def_cache_size =`*user-defined-type-information-buffer-size*`]`#	
24	`[set pd_routine_def_cache_size =`*routine-definition-information-buffer-size*`]`#	
25	`[set pd_registry_cache_size =` *registry-information-buffer-size*`]`#	
26	`[set pd_sds_shmpool_size =` *single-server-shared-memory-size*`]`#	Shared memory
27	`[set pd_rpc_trace = Y | N]`#	RPC trace information
28	`[set pd_rpc_trace_name = "`*name-for-RPC-trace-collection-files*`"]`#	
29	`[set pd_rpc_trace_size =` *RPC-trace-collection-file-size*`]`#	
30	`[set pd_module_trace_max =`*maximum-number-of-module-traces-that-can-be-stored*`]`#	Troubleshooting information
31	`[set pd_module_trace_timer_level = 0 | 10 | 20]`#	
32	`[set pd_pth_trace_max =` *maximum-number-of-stored-communication-traces*`]`#	
33	`[set pd_max_add_dbbuff_no =` *maximum-global-buffers-count-for-dynamic-addition*`]`#	Global buffers
34	`[set pd_max_add_dbbuff_shm_no =` *maximum-shared-memory-segments-count-for-dynamic-addition*`]`#	
35	`[set pd_max_temporary_object_no =` *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*`]`#	Temporary tables
36	`[set pd_audit_def_buffer_size =` *security-audit-information-buffer-length*`]`#	Security
37	`[set pd_plugin_ixmk_dir = "`*index-information-file-creation-directory-name*`"` *or* `"`*index-information-file-creation-HiRDB-file-system-area-name*`"]`#	Delayed batch creation of plug-in index
38	`[set pd_java_stdout_file = "`*Java-virtual-machine-standard-output-or-standard-error-output-destination-file*`"]`#	Java
39	`[set pd_java_castoff = Y | N]`#	
40	**[set pd_log_dual = Y | N]**#	System log files
41	`[set pd_log_remain_space_check = warn | safe]`#	
42	`[set pd_log_auto_unload_path = "`*unload-log-file-output-directory*`"` *[,*`"`*unload-log-file-output-directory*`"]...]`	
43	`[set pd_log_auto_unload_restart = Y | N]`#	
44	`[set pd_log_singleoperation = Y | N]`#	
45	`[set pd_log_rerun_reserved_file_open = Y | N]`#	
46	`[set pd_log_rerun_swap = Y | N]`#	
47	`[set pd_log_swap_timeout =` *wait-time-for-completion-of-system-log-file-swapping*`]`#	
48	`[set pd_log_unload_check = Y | N]`#	
49	`[set pd_log_max_data_size =` *log-input/output-buffer-size*`]`#	
50	`[set pd_log_write_buff_count =` *log-output-buffer-sectors-count*`]`#	

No.	Format	Operand category
51	[set pd_log_rec_leng = *system-log-file−record-length*]#	
52	[set pd_log_rollback_buff_count = *rollback-log-input-buffer-sector-count*]#	
53	[set pd_log_auto_expand_size = *extension-amount-per-system-log-file-extension-trigger[,extension-limit*]]#	
54	**[set pd_spd_dual = Y | N]#**	Synchronization point dump files
55	[set pd_spd_assurance_msg = Y | N]#	
56	[set pd_spd_assurance_count = *number-of-guaranteed-valid-generations*]#	
57	[set pd_spd_reduced_mode = *reduced-mode-operation-option*]#	
58	[set pd_spd_reserved_file_auto_open = Y | N]#	
59	[set pd_spd_max_data_size = *synchronization-point-dump-file-buffer-size*]#	
60	[set pd_log_sdinterval = *system-log-output-volume*[,*interval*]]#	
61	**set pd_sts_file_name_1 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"**	Server status files
	[set pd_sts_file_name_2 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_3 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_4 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_5 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_6 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_7 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
62	[set pd_sts_initial_error = stop | continue | excontinue]#	Server status files (when an error occurs)
63	[set pd_sts_singleoperation = stop | continue]#	
64	[set pd_sts_last_active_file = "*logical-file-name*"]	
65	[set pd_sts_last_active_side = A | B]	
66	**pdwork -v "*HiRDB-file-system-area-name*"[,"*HiRDB-file-system-area-name*"]...**	Work table files
67	**{{pdlogadfg -d sys -g *file-group-name* [ONL]}}**	System log file configuration
68	**{{pdlogadpf -d sys -g *file-group-name* -a "*system-log-file-name*" [-b "*system-log-file-name*"]}}**	
69	**{{pdlogadfg -d spd -g *file-group-name* [ONL]}}**	Synchronization point dump file configuration
70	**{{pdlogadpf -d spd -g *file-group-name* -a "*synchronization-point-dump-file-name*" [-b "*synchronization-point-dump-file-name*"]}}**	

No.	Format	Operand category
71	`{{[pdplgprm -n` *plug-in-name* `[-s` *shared-memory-size*`]]}}`	Plug-ins

#: If this operand is omitted, the value specified in the same operand in the server common definition is used. However, for the following operands, the value specified for the same operand in the system common definition, rather than the server common definition, is used:

- `pd_cwaittime_wrn_pnt`
- `pd_uap_exerror_log_use`

5.2 Operand explanations

5.2.1 Operands related to processes

1) pd_process_count = *resident-processes-count*[*,resident-processes-count-at-server-startup*]

~<unsigned integer>((0-3000))

resident-processes-count

Specifies the number of processes that can be made resident in the single server. A resident process is a process that is activated at the time the server is started.

Advantage

By activating the processes used by transactions that can be processed concurrently by the single server at the time of system startup and keeping them resident, the process startup time can be reduced even when new transactions are entered. However, HiRDB startup will take longer.

Specification guidelines

- The value to be specified is determined on the basis of the process private area of the single server's server process and the real memory size of the processor. For details about the process private area of server processes, see the *HiRDB Version 9 System Operation Guide*.
- If a multi front-end server configuration is used and the `pd_max_bes_process` or `pd_max_dic_process` operand is also specified, specify for the `pd_process_count` operand a value that satisfies the following condition:

  Value of pd_process_count $\leq$ (*value of pd_max_bes_process or value of pd_max_dic_process*)
- The value specified for this operand must be no more than the maximum number of processes that can be activated for the single server (*value of pd_max_users*).

Tuning the specified value

For details about how to tune the resident processes count, see the *HiRDB Version 9 System Operation Guide*.

Notes

- Because the number of resident processes has a direct effect on the availability of memory space and on the CPU, specifying an unnecessarily large number might prevent HiRDB from starting or might degrade the server machine's processing performance.
- If more processes than the resident process count are needed, additional processes are dynamically started, up to the maximum processes count allowed. However, depending on the value specified for the `pd_max_server_process` operand, it might not be possible to start all of the processes indicated by the maximum processes count.

Operand default

When this operand is omitted (or `0` is specified), the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the maximum number of processes.

resident-processes-count-at-server-startup

Specifies the number of processes that can be made resident during HiRDB startup.

It is common for resident processes to be activated during HiRDB startup. If there are many resident processes, the amount of time required for HiRDB startup increases proportionately. For example, it takes approximately 1 second to activate a process on a server machine with a 100-MIPS performance rating.

The differences in processing that result depending on whether a resident processes count at server startup is specified are as follows:

- When there is no specification of a resident processes count at server startup (when, for example, `pd_process_count = 500` is specified)

  All 500 resident processes are activated during HiRDB startup, and HiRDB will not start until they are all activated. In this example, it would take a 100-MIPS server machine about 500 seconds to activate all the resident processes during HiRDB startup.
- When a resident processes count at server startup is specified (when, for example, `pd_process_count = 500, 50` is specified)

Some of the resident processes (50 in this case) are activated during HiRDB startup, and the remaining resident processes are activated after HiRDB startup. HiRDB can be started as long as the specified number of resident processes are activated. In this example, it would take a 100-MIPS server machine about 50 seconds to activate 50 resident processes during HiRDB startup. The remaining 450 resident processes (in this example) would be activated after HiRDB startup.

Advantage

The amount of time required for HiRDB startup is reduced. Use this option when you want to reduce the HiRDB startup time as much as possible, such as when you are using the system switchover facility.

Specification guideline

Specify a value equal to the number of processes that will be required immediately after HiRDB startup is completed.

Notes

When you specify a resident processes count at server startup, recheck the value in the `PDCWAITTIME` operand of the client environment definition.

If more UAPs than the resident processes count at server startup are to be executed immediately following HiRDB startup, transaction processing might not be performed until after the remaining resident processes have been activated. Therefore, if the value specified in the `PDCWAITTIME` operand of the client environment definition is small, it might not be possible to process some UAPs due to timeouts. For details about the `PDCWAITTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.

2) pd_server_cleanup_interval = *interval-for-stopping-nonresident-server-processes*

~<unsigned integer>((0-1440)) (minutes)

Specifies in minutes the interval for checking for nonresident server processes in HiRDB that are to be stopped. The facility for stopping nonresident server processes is applied when the number of executing server processes exceeds the number of processes that can be made resident (value specified by the `pd_process_count` operand). The number of server processes that the facility stops is computed automatically by HiRDB.

Advantages

This facility improves the utilization rates of the memory and of process resources because it increases the number of nonresident processes that can be reused when the workload (number of server processes that are executing) peaks.

Specification guidelines

- For example, if there is only one hour a day during which peak workload occurs and the intervals between peaks within that hour are approximately two minutes apart, specify `2` for this operand.
- This facility does not have any effect if the number of server processes that execute concurrently during peak periods is always fewer than the number of resident processes. In this case, omit this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Tuning the specified value

Collect statistical information on system operations for each server for one week. Determine the workload peaks from the number of server processes being serviced (`# OF PROCESSES ON SERVICE`). If that peak value exceeds the currently set number of processes that can be made resident (value specified by the `pd_process_count` operand), determine the interval between individual peaks and set that number of minutes.

However, if there are ample resources, such as memory and CPU, in the server machine, adding the shortfall in the number of processes to the number of resident processes (in other words, increasing the value of the `pd_process_count` operand) is more effective in improving performance than specifying the `pd_server_cleanup_interval` operand.

Note

When this operand is omitted or `0` is specified, the system checks every 10 seconds for nonresident server processes that are waiting to be serviced and stops any such processes that are found.

3) pd_max_ard_process = *asynchronous-READ-process-count*

~<unsigned integer>((0-256))

Specify this operand if you use the asynchronous READ facility. For this operand, specify the number of processes necessary for asynchronous READ operations. For details about the asynchronous READ facility, see the *HiRDB Version 9 Installation and Design Guide*.

Condition

A value of `1` or greater must be specified for the `-m` option of the `pdbuffer` operand.

Specification guidelines

- Specify `0` or `1`. However, if a value between 2 and 256 is specified for the `-m` option of the `pdbuffer` operand, specify the same value as the `-m` option value. If a value greater than 256 is specified for the `-m` option, specify the same value as the number of disk devices that store RDAREAs and system files or `256`.
- Increasing the value of this operand can shorten the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. Decreasing the value of this operand might increase the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. This is because asynchronous READ processes might have to wait for processing completion.
- Because a number of processes equaling *value of this operand* × *server count* are started, determine a value for this operand by taking resources (shared memory and message queue) into consideration. For details about estimating shared memory and message queue sizes, see the *HiRDB Version 9 Installation and Design Guide*.

Tuning the specified value

For details about how to tune the specification value (number of asynchronous READ processes), see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Operand rule

If you specify `0` for this operand, the asynchronous READ facility is not used.

Relationship to other operands

If you change the value of this operand, re-evaluate the value of the `pd_max_server_process` operand.

Effects on individual estimation formulas

If the value of the `pd_max_ard_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formula 2* under *Formulas for shared memory used by a single server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*

4) pd_dfw_awt_process = ***number-of-parallel-writes-for-deferred-write-processing***

~<unsigned integer>((2-255))

Specify this operand when you use the *facility for parallel writes in deferred write processing* for all buffer pools. Specify for this operand the number of processes to be processed in parallel. Increasing the number of processes can shorten the write processing time. For details about the facility for parallel writes in deferred write processing, see the *HiRDB Version 9 Installation and Design Guide*.

Specification guidelines

Specify `2`, which is the smallest value that enables the facility for parallel writes in deferred write processing. Furthermore, to determine the value for this operand, see *Tuning deferred write processing* in the *HiRDB Version 9 System Operation Guide*.

Note

Specifying the facility for parallel writes in deferred write processing increases the number of processes and consequently raises the CPU usage rate.

Effects on individual estimation formulas

If the value of the `pd_dfw_awt_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*
- *Formula 5* under *Formulas for shared memory used by a single server*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*

5.2.2 Operands related to work tables

5) pd_work_buff_mode = each | pool

Specifies the method of allocating buffers when HiRDB creates tables.

`each`: Allocate a buffer for each work table.

`pool`: Allocate a buffer pool for each server process.

Specification guidelines

- Normally, `pool` is specified. `pool` is the appropriate specification when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.
- When the size of the process private area that can be used for work table buffers is predetermined, specify `pool`. When `pool` is specified, HiRDB efficiently allocates work table buffers to work tables.

  In such a case, the process private area is occupied on the basis of the value specified in `pd_work_buff_size`, and input/output operations on work tables are buffered in that pool. Therefore, the process private memory is occupied only to the extent of the value specified in `pd_work_buff_size`.

Notes

If `each` is specified for this operand, the amount of memory used might increase.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `pool` is assumed (if `v6compatible` is specified in the `pd_sysdef_default_option` operand, `each` is assumed).

Effects on individual estimation formulas

If the value of the `pd_work_buff_mode` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Considerations when migrating to 64-bit mode*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*

6) pd_work_buff_size = *work-table-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

Specifies in kilobytes the size of buffers for work tables to be created by HiRDB.

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
Advantage	A large work table buffer size reduces the number of I/O operations associated with data manipulation, which means that the execution time of SQL statements that use work tables is also reduced. However, because each server's process private memory is used, you must also take into account the overall size of the system memory (real memory and virtual memory) when you specify this option. If `pd_work_buff_mode = each` is specified, the memory size to be allocated is *value of pd_work_buff_size* × *required number of work tables*. Therefore, specifying an unnecessarily large value might cause a virtual memory shortage for other processes.	

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
Application criterion	Specify `pd_work_buff_mode` = `pool` when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.	
Specification guidelines	• Specify the size of the buffer to be allocated for one work table. • If a value greater than the work table memory capacity is specified for the work table buffer size, input/output to the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the work table memory capacity: *Work table memory capacity = Applicable work table size*# $\div$ 2	• Specify the size of the buffer pool to be allocated for the entire server process. • Specify a value between `4352` and `5120` when a large volume of data is to be retrieved or when manipulations such as join, `ORDER BY`, and `GROUP BY` are to be performed. Specifying such a value increases the unit of sorting input/output, thus reducing the sort time. • If a large value is specified for the work table buffer size, with the total work table memory capacity for each SQL statement as the upper limit, input/output operations on the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the total work table memory capacity for each SQL statement: *Total work table memory capacity per SQL statement* = $a \times b + c \times d$
Notes	• When multiple users execute processes concurrently or when an SQL statement that uses multiple work tables is executed, a buffer of the specified size is allocated for each work table. Consequently, specifying a large value might result in a memory shortage. • If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.	• If the value specified for the work table buffer size is smaller than the number of work tables to be used by each SQL statement, the processing time might become longer than when `each` is specified. Specifically, specify a value that is at least equal to *maximum number of work tables for each SQL statement* $\times$ 128. The following formula can be used to determine the maximum number of work tables for each SQL statement: *Maximum number of work tables for each SQL statement* = $b + d$ • If the specified buffer size is too large to be allocated in the system, the allocation of process private memory fails and the server cannot start up.
Operand rule	Specify a multiple of `128`. If the value is not a multiple of `128`, it is rounded up to the next multiple of `128`.	• Specify a multiple of `128`. If the value is not a multiple of `128`, it is rounded up to the next multiple of `128`. • Specify at least `384`. If a value that is smaller than `384` is specified, it is rounded up to `384`.
Default value	If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `128` is assumed.	If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the following value is assumed: • 32-bit mode: `384` • 64-bit mode: `5120`

a:

↑ {Capacity of work table (for storing column information)# **(kilobytes)** $\div$ 2} $\div$ 128 ↑ $\times$ 128

b:

Maximum number of work tables (for storing column information)#

c:

↑ {Capacity of work table (for storing positional information)# (kilobytes) $\div$ 2} $\div$ 128 ↑ $\times$ 128

d:

Maximum number of work tables (for storing positional information)#

#: For details about how to determine these values, see the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_work_buff_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*

7) pd_work_buff_expand_limit = *work-table-buffer-expansion-limit*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

The size of the work table buffer to be created by HiRDB is specified by the `pd_work_buff_size` operand. Specify the `pd_work_buff_expand_limit` operand if you want to automatically expand a work table buffer when the space in this buffer becomes insufficient. The work table buffer is expanded up to the size specified by this operand.

For example, when the following values are specified for the operands, a 1,024-kilobyte work table buffer is normally allocated. When this size becomes insufficient, the work table buffer is expanded up to 2,048 kilobytes.

- `pd_work_buff_size = 1024`
- `pd_work_buff_expand_limit = 2048`

HiRDB expands a work table buffer in the following cases:

- The necessary work table buffer cannot be allocated when hash execution is applied to an execution method that uses hash join or subquery hash as the joining method.
- A 128-kilobyte work table buffer allocated to each work table becomes insufficient when multiple work tables are concurrently used.

Condition

The `pd_work_buff_mode` operand must be omitted, or `pool` must be specified for it.

Advantage

You can prevent a work table buffer shortage (too small a value specified for the `pd_work_buff_size` operand) from causing UAP errors.

Notes

- A work table buffer is not expanded when either of the following conditions is satisfied:
  - The `pd_work_buff_expand_limit` operand is not specified.
  - `pd_work_buff_expand_limit` operand value ≤ `pd_work_buff_size` operand value
- If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.

Operand rule

Specify a multiple of 128. If a value other than a multiple of 128 is specified, it is automatically rounded up to a multiple of 128.

Relationship to other operands

When a work table buffer is expanded for the first time in a single server process, the `KFPH29008-I` message is output. Note that you can use the `pd_work_table_option` operand to suppress this message output.

Note

After a work table buffer has been expanded, when the number of work tables being used by the applicable server process goes to zero, the expanded work table buffer is released. The number of work tables being used can go to zero in the following cases:

- All cursors that were being used are closed. (In this case, the number of work tables being used might not go to zero.)
- A transaction is normally terminated or cancelled when a holdable cursor is not being used.
- A UAP is disconnected from HiRDB when a holdable cursor is being used.

Remarks

Hash join, subquery hash execution is applied in the following cases:

- *Application of optimizing mode 2 based on cost* and *hash join, subquery hash execution* are specified in the `pd_additional_optimize_level` operand, the `PDADDITIONALOPTLVL` operand of the client environment definition, or the `ADD OPTIMIZE LEVEL` operand of the SQL compile option.
- `HASH` is specified for the SQL optimization specification of the joining method inside an SQL statement.
- `HASH` is specified for the SQL optimization specification of the subquery execution method inside an SQL statement.

Effects on individual estimation formulas

If the value of the `pd_work_buff_expand_limit` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*

5.2.3 Operands related to system monitoring

8) pd_watch_pc_client_time = ***maximum-client-request-wait-time***

~<unsigned integer>((0-65535)) (seconds)

Specifies in seconds the maximum amount of time for a server to wait for the next request from a HiRDB client after the HiRDB server returns a response to a request from a Windows-compatible HiRDB client.

If no request comes from the HiRDB client within the specified amount of time, it will be assumed that an error occurred at the client and the connection between the server and the client will be terminated forcibly. No notice of disconnection is sent to the HiRDB client in such a case.

The time that is monitored is the period between `CONNECT` and `DISCONNECT` (that is, the non-transaction status time), excluding the period between SQL execution startup and `COMMIT` or `ROLLBACK`.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `3600` is assumed.

Notes

- When `0` is specified, the server waits indefinitely for the next request from the HiRDB client.
- If a small value (up to `600`, for example) is specified for this operand, the HiRDB client might detect *server down* during SQL execution and might not terminate correctly.
- For a UNIX edition of a HiRDB client (including a Linux edition of a HiRDB client), time is not monitored, regardless of the value specified for this operand. To monitor time for a UNIX edition of a HiRDB client, specify the `PDSWATCHTIME` client environment definition of the HiRDB client.

Relationship to client environment definition

The value of this operand can be modified for each client. To do so, the `PDSWATCHTIME` operand must be specified in the client environment definition. For details about the `PDSWATCHTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.

9) pd_spd_syncpoint_skip_limit = ***maximum-number-of-skipped-synchronization-point-dumps***

~<unsigned integer>((0, 2-100000))

If an event such as an endless loop occurs in a UAP, acquisition of synchronization point dumps might be skipped. If several synchronization point dumps are not acquired (instead, they are skipped), there will be an increase in the number of overwrite-disabled system log files, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

If HiRDB is forcibly terminated or terminates abnormally when the number of system log files that cannot be overwritten has reached one-half or more of all system log files, a shortage of system log files occurs during rollback processing when HiRDB is restarted. In this case, HiRDB cannot be restarted unless new system log files are added. Any such restart processing will take a longer time than normal.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction). When the number of skipped synchronization point dumps reaches the specified operand value, the target transaction is cancelled forcibly and rolled back. This is called the *skipped effective synchronization point dump monitoring facility*.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

Advantage

Specifying this operand provides a means of dealing with endless loops in UAPs.

Specification guidelines

- Normally, specify `0` for this operand. When `0` is specified, HiRDB computes the upper limit for the skip count. If specifying `0` causes a problem or the `KFPS02101-I` message is issued, change the value of this operand. For guidelines on the value to specify, see the *HiRDB Version 9 System Operation Guide*.
- If the specified value is too large, all system log files might be placed in overwrite disabled status. If this happens, HiRDB terminates abnormally.
- If the specified value is too small, the number of transactions that are forcibly rolled back might increase.
- Specify this value taking into account the value of `pd_log_sdinterval` and the number of times synchronization point dumps are acquired when the transaction with the longest execution time and largest log output volume is processed.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the skipped effective synchronization point dump monitoring facility is not used.

Notes

- The monitoring provided for by the specification of this operand begins the first time a synchronization point dump is collected after HiRDB is started (including a restart).
- When this operand is specified, UAPs that are being executed in the no-log mode are also monitored. If the processing of a UAP being executed in the no-log mode is cancelled, the database cannot be automatically recovered, and thus RDAREAs are placed in the error shutdown state. Therefore, when specifying the upper limit, account also for the size of the system logs that are output from other transactions inside the server during the processing of UAPs that are executed in the no-log mode.
- If a transaction is delayed due to a high workload, the skip count increases because a synchronization point dump cannot be acquired until the delayed transaction is completed.
- If the skip count exceeds the upper limit, any process executing a transaction is forcibly terminated. In this case, the rollback logs, which are the logs output by these transactions, are output after the forced termination.
- The `pdload`, `pdmod`, `pdrorg`, `pdexp`, `pddbst`, `pdgetcst`, `pdrbal`, `pdvrup`, `pdmemdb`, and `pdextfunc` commands are not monitored by this facility.

10) pd_dfw_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*

~<unsigned integer>((0-100000))

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing is completed, acquisition of the synchronization point dump is skipped. This is because acquisition of the synchronization point dump is delayed by the deferred write processing, and the number of update buffers output by the synchronization point exceeds the number of update buffers that can be output within the synchronization point dump acquisition interval.

If more than one synchronization point dump is skipped, there will be an increase in the number of system log files that cannot be overwritten, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction) due to deferred write processing.

When the number of skipped synchronization point dumps due to deferred write processing reaches the specified operand value, HiRDB determines the maximum number of update buffers in such a manner that acquisition of a synchronization point dump can be completed within the synchronization point dump acquisition interval. If the maximum number of update buffers is exceeded, HiRDB then outputs the oldest update buffer and limits the total size of update buffers at a synchronization point. This is called the *update buffer size restriction facility*.

Advantage

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing has terminated, the unit terminates abnormally. This operand enables such abnormal termination of the unit to be avoided.

Specification guidelines

Normally, you will omit this operand. If you wish to prevent unit abnormal termination caused by the occurrence of a synchronization point before termination of deferred write processing within the synchronization point dump acquisition interval, specify 1.

If an acceptable number of times synchronization point dumps can be skipped can be determined in advance, such as from the size of the log information, specify that value.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `0` is assumed.

Operand rules

If `0` is specified in this operand, HiRDB does not use the update buffer size restriction facility.

Notes

The following notes explain the period of effectiveness of the update buffer size restriction facility. The period of effectiveness means the interval between issuance of the `KFPH23035-I` message and issuance of the `KFPH23036-I` message.

- If the number of update buffers exceeds the maximum number of update buffers determined by HiRDB, update buffers are output after update processing has executed; this degrades the update processing throughput. You can use the following formula to obtain the maximum number of update buffers determined by HiRDB:

(*synchronization point dump interval* ÷ *unit value of WRITE*#)
× (1 - (*amount of log information from the previous synchronization point dump to the pre-synchronization point*)
× (*number of buffer sectors in buffer pool* ÷ *total number of buffer sectors in buffer pool that was updated at the synchronization point*)

#: For details about the unit value of `WRITE`, see the *HiRDB Version 9 System Operation Guide*.

- If the deferred write trigger is specified in the `pd_dbbuff_rate_updpage` or `pdbuffer -y` operand, and each operand value becomes greater than the maximum number of update buffers determined by HiRDB, the maximum number of update buffers determined by HiRDB is changed to the number of update buffers that triggers deferred write processing.

  The value of the `pdbuffer -w` operand is adjusted automatically so that up to the maximum number of update buffers is output for each buffer.
- A skipped synchronization point dump is detected during update buffer output processing at a synchronization point. Therefore, the update buffer size restriction facility might be enabled after a synchronization point dump is skipped and an error message is displayed.
- Normally, when the parallel writes facility is used, there is one output request per synchronization point for each parallel WRITE process for deferred write processing during synchronization point processing. However, if the update buffer restriction facility is used, there will be more than one output request in order to facilitate early detection of skipped synchronization point dumps.

5.2.4 Operands related to SQL runtime warning output facility

11) pd_cwaittime_wrn_pnt = *output-condition-for-SQL-runtime-warning-information (% specification)* | *output-condition-for-SQL-runtime-warning-information (time specification)*

Specify this operand when using the SQL runtime warning output facility. You can specify this operand using one of the following two methods:

- Specifying a percentage
- Specifying a time duration

For details about the SQL runtime warning output facility, see the *HiRDB Version 9 System Operation Guide*.

***output-condition-for-SQL-runtime-warning-information (% specification)*: ~<unsigned integer>((0-99)) or <unsigned decimal number>((0-99.999999)) (%)**

Specifies the condition for outputting SQL runtime warning information of the SQL runtime warning output facility as a percentage of the maximum client wait time (value of the `PDCWAITTIME` operand in the client environment definition). After SQL execution, HiRDB checks the SQL execution time. If this SQL execution time is determined to exceed the time specified by this operand, the following warning information is output. This is called the *SQL runtime warning output facility*.

- SQL runtime warning information file
- Warning message (`KFPA20009-W`)

Operand specification methods

Specify this operand as a percentage (%) of the value of the `PDCWAITTIME` operand. For example, if `100` (seconds) is specified for the `PDCWAITTIME` operand and `90` (%) is specified for the `pd_cwaittime_wrn_pnt` operand, HiRDB checks the SQL execution time after SQL execution. If the determined SQL execution time is 90 seconds or longer but less than 100 seconds, the warning information is output.

Example

```
PDCWAITTIME = 100
pd_cwaittime_wrn_pnt = 90
```

***output-condition-for-SQL-runtime-warning-information (time specification)*: ~<unsigned decimal number>((0-65534.999999))sec (seconds)**

Specifies the condition for outputting SQL runtime warning information of the SQL runtime warning output facility as a time duration. After SQL execution, HiRDB checks the SQL execution time. If this SQL execution time is determined to exceed the time specified by this operand, the following warning information is output. This is called the *SQL runtime warning output facility*.

- SQL runtime warning information file
- Warning message (`KFPA20009-W`)

Operand specification method

Specify the time duration (in seconds) to be used as the output trigger. (You can specify up to the sixth decimal.) Add `sec` to the specified value.

Example

```
pd_cwaittime_wrn_pnt = 0.001sec
```

The following explanation is the same for both the percentage and time duration specification.

Operand rule

If `0` or `0sec` is specified in this operand, no warning information is output. (The SQL runtime warning output facility is not used.)

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, the following value is assumed:

- *output-condition-for-SQL-runtime-warning-information (% specification)*: `0`
- *output-condition-for-SQL-runtime-warning-information (time specification)*: `0sec`

Relationship to client environment definition

You can change the value of this operand for each client. To change it for each client, specify the `PDCWAITTIMEWRNPNT` operand of the client environment definition. For details about the `PDCWAITTIMEWRNPNT` operand, see the *HiRDB Version 9 UAP Development Guide*.

Relationship to other operands

This operand is related to the following operands:

- `pd_cwaittime_report_dir`
- `pd_cwaittime_report_size`

5.2.5 Operand related to the facility for output of extended SQL error information

12) pd_uap_exerror_log_use = YES | NO

Specifies whether to use the facility for output of extended SQL error information. For details about this facility, see the *HiRDB Version 9 UAP Development Guide*.

`YES`:

The facility for output of extended SQL error information is used. SQL error information is output to a client error log file and the SQL error report file.

`NO`:

The facility for output of extended SQL error information is not used.

Specification guidelines

If you manage SQL error information centrally, or output the SQL statements and the parameter information resulting from an error, we recommend that you specify `YES`.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `NO` is assumed.

Relationship to client environment definition

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDUAPEXERLOGUSE` operand in the client environment definition. If both this operand and the `PDUAPEXERLOGUSE` operand in the client environment definition are specified, the `PDUAPEXERLOGUSE` operand takes precedence.

For details about the `PDUAPEXERLOGUSE` operand, see the *HiRDB Version 9 UAP Development Guide*.

Effects on individual estimation formulas

If the value of the `pd_uap_exerror_log_use` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the size of the memory required to use the facility for output of extended SQL error information* under *Estimating the memory size required for a HiRDB/Single Server*

5.2.6 Operands related to lock

13) pd_lck_pool_size = ***server-lock-pool-size***

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-2000000))**
- 64-bit mode: **((1-2000000000))**

Specifies in kilobytes the size of the shared memory area to be used by the single server for locking (lock pool).

The area specified by this operand is used for tables for managing locked resources. Note the following relationship between the number of tables for managing locked resources, the number of lock requests, and the lock pool size:

Number of tables for managing locked resources = number of lock requests = size of lock pool × coefficient

Specification guidelines

- In the 32-bit mode, the amount of lock pool space required for six lock requests is 1 kilobyte.
- In the 64-bit mode, the amount of lock pool space required for four lock requests is 1 kilobyte.
- The following formulas can be used to determine the value to be specified for this operand:

HiRDB type	Formula (kilobytes)
HiRDB/Single Server (32-bit mode)	↑↑ *a* ÷ *value of pd_lck_pool_partition* ↑ ÷ 6 ↑ × *value of pd_lck_pool_partition*

HiRDB type	Formula (kilobytes)
HiRDB/Single Server (64-bit mode)	↑↑ *a* ÷ *value of pd_lck_pool_partition* ↑ ÷ 4 ↑ × *value of pd_lck_pool_partition*

a: Total number of transaction lock requests to be executed concurrently by the single server. The number of lock requests depends on the SQL. For details about how to determine the total number of lock requests, see *D. Determining the Number of Locked Resources*.

Note

When you execute `DROP TABLE` or `DROP SCHEMA` in a definition SQL, it is especially important to have already determined in advance an appropriate value for this operand.

Tuning the specified value

See the usage rate for the locked resources management table (`% OF USE LOCK TABLE`) displayed in the statistical information on system operation by the statistics analysis utility. If the maximum usage rate equals or exceeds 80%, it is recommended that the operand's value be increased in preparation for future database expansion. If the maximum usage rate does not exceed 10%, it is recommended that the operand's value be decreased to conserve shared memory space.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the appropriate value shown below:

- For 32-bit mode: `16000`
- For 64-bit mode: `32000`

However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `1024`.

Note

- If the value specified for this operand is too small, an SQL statement might return an error.
- Do not specify a larger value than necessary in this operand. A large value will increase the size of the shared memory used by HiRDB, which might cause a shortage of shared memory and prevent HiRDB from starting.
- If you do an all-item search of a table that contains many rows while locking is in units of rows, this operand's value will need to be increased to reflect the large number of items, which will increase the amount of memory required. Instead, consider making the following adjustments in the UAP:
  - ● Acquire locks in table units
  - ● If you can use the unlocked search facility, perform searches unlocked
  - ● Narrow the search conditions, and divide the processing into several transactions

Relationship to other operands

This operand is related to the `pd_lck_pool_partition` operand.

14) pd_lck_pool_partition = *per-server-lock-pool-partition-count*

~<unsigned integer>((1-5000))

Specifies the number of lock pool partitions to be used in locking by the single server when distributing lock processing.

For details about distributing lock processing, see the *HiRDB Version 9 System Operation Guide*.

Tuning the specified value

For details about how to tune the number of lock pool partitions, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `1`.

Notes

- When the value set in this operand is too large, the size of the shared memory assigned to each lock pool partition becomes smaller, which might result in not enough lock pool partition capacity, causing SQL to return an error. Lock processing also takes time, so system performance declines. If this becomes an issue, specify a smaller value in this operand.

- The lock pool size must be at least 1 kilobyte. If a value larger than the value of `pd_lck_pool_size` is specified, the `KFPS00421-W` message will be issued and the value of `pd_lck_pool_size` will be assumed for this operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_pool_size`
- `pd_lck_deadlock_check_interval`

Effects on individual estimation formulas

If the value of the `pd_lck_pool_partition` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Single Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*

15) pd_lck_until_disconnect_cnt = *total-number-of-tables-and-RDAREAs-to-be-locked-until-disconnect-specification*

~<unsigned integer>((0-140000))

Specifies the number of resources to be locked for the tables and RDAREAs that are to be held across transactions. Based on the value specified for this operand, blocks for which lock with `UNTIL DISCONNECT` is specified for the tables and RDAREAs are allocated in the shared memory.

Specification guidelines

Normally, this operand need not be specified. Specification of a value other than the default value might be necessary in the following cases:

- When the number of utilities to be executed concurrently increases
- When a holdable cursor is used
- When the local buffer specified in the `pdlbuffer` operand is used
- When an SQL session-specific temporary table is used

For details about how to estimate the specification value for this operand, see *C.5 Formula for determining total number of tables and RDAREAs per server locked with UNTIL DISCONNECT specified (pd_lck_until_disconnect_cnt)*.

Tuning the specified value

If the value specified for this operand is small, a transaction might roll back or a utility might terminate abnormally with return code `8`. In such cases, the message `KFPA11914-E` or `KFPH28001-E` is output. If this occurs, increase the value of this operand.

When the value of this operand is increased, the amount of required memory space increases proportionately. The required memory size can be expressed as follows: *value of this operand* × 48 (64 in the 64-bit mode) bytes.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `256`.

Effects on individual estimation formulas

If the value of the `pd_lck_until_disconnect_cnt` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Determining the number of records in a synchronization point dump file*
- *Formula 2* under *Formulas for shared memory used by a single server*

16) pd_max_open_holdable_cursors = *maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*

~<unsigned integer>((16-1024))

When you use holdable cursors for a table for which a `LOCK` statement with `UNTIL DISCONNECT` specification is not executed, this operand specifies the maximum number of holdable cursors that can be concurrently open for each transaction.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `16`.

Note

Specifying a value other than the default value for this operand increases the amount of shared memory used.

Relationship to other operands

The values specified for this operand and the following operands are used for computing the shared memory size for lock servers. For a 32-bit mode HiRDB system, if the values specified for these operands are too large, the shared memory size of the lock servers exceeds 2 GB, and as a result, HiRDB might not start. Therefore, adjust the values specified for these operands so that the shared memory size of the lock servers does not exceed 2 GB.

- `pd_max_access_tables`
- `pd_max_users`
- `pd_lck_hash_entry`
- `pd_lck_pool_size`

For details about shared memory, see the *HiRDB Version 9 Installation and Design Guide*.

17) pd_lck_hash_entry = *lock-pool-hash-entry-count*

~<unsigned integer>((0-2147483647))

Specifies the number of hash table entries to be used in the lock pool. According to the value specified here, HiRDB allocates a lock pool in the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand.

Consider specifying a value for this operand in the following cases:

- If you do not wish to change the shared memory size if possible when upgrading to version 06-02 or later, specify `11261`. In this case, the same number of hash entries is allocated as in the earlier version, and the hash table size inside the lock pool remains the same as before.
- It is possible to improve performance by specifying in this operand a value greater than the recommended value shown below. However, specifying a value greater than variable *a* (also shown below) will not improve performance over the case in which *a* is specified.

  The recommended value is as follows:

  Recommended value = Largest prime number not exceeding MAX($\uparrow a \div 10 \uparrow$, 11,261)

Variable	Formula for computing the variable
a	(*pd_max_users* + 3) $\times$ (*pd_max_access_tables value* + 14) + *pd_lck_pool_size* $\times$ *c*
c	6 for the 32-bit mode; 4 for the 64-bit mode

Operand rules

- If this operand and the `pd_lck_hash_entry` operand of the server common definition are both omitted or `0` is specified in this operand, HiRDB calculates a recommended value for the server. (However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default for this operand is `11261`.)
- If a non-zero value or a non-prime number is specified for this operand, HiRDB assumes that the largest prime number not exceeding the specified value has been specified.

Note

If the value specified for this operand is too small, a hash entry shortage might occur, resulting in performance degradation. If this operand is omitted, neither hash entry shortage nor performance degradation due to hash entry shortage occurs.

18) pd_dbsync_lck_release_count = *global-buffer-lock-release-interval-during-synchronization-point-processing*

~<unsigned integer>((0, 100-1073741824))

Specifies an interval for unlocking global buffers, when global buffer locking occurs during synchronization point processing.

During synchronization point processing, search processing occurs on the buffers (update buffers) and must be applied to the disk. Normally, global buffers are unlocked at a specific interval during search processing on the update buffers.

For example, if 100 is specified in this operand, a global buffer is unlocked once when search processing on 100 sectors (global buffer sectors) is completed. After that, the global buffer is locked again and search processing is resumed. In this example, unlocking occurs once every 100 sectors.

Advantage

By specifying this operand, you can adjust the global buffer lock time during synchronization point processing. When a small value is specified in this operand, the global buffer lock time becomes short and transaction performance might improve during synchronization point processing.

To obtain the global buffer pool lock time, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Buffer pool lock time during synchronization point processing (SYNCL)`.

Specification guidelines

Normally, there is no need to specify this operand. Consider specifying this operand when both the following conditions apply:

- Transaction performance drops during synchronization point processing.
- A large number of buffer sectors is specified in the `-n` option of the `pdbuffer` operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `10000`.

Operand rules

- If the specified value is in the range 1 to 99, 100 is set automatically.
- If 0 is specified, global buffers are locked until update buffer search processing is completed.

Notes

If a small value is specified in this operand, the update buffer search time increases due to interrupts from other transactions and CPU usage rises. The global buffers updated during that time are also output during synchronization point processing. Therefore, the number of update buffers to be output during synchronization point processing increases. To obtain the number of update buffers to be output during synchronization point processing, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Number of synchronization point output pages (SYNCW)`. As you increase the value of this operand, the lock time for a global buffer to determine the buffer to be output at a synchronization point will increase. For this reason, lock contention grows substantially during synchronization point processing, which might affect transaction performance.

5.2.7 Operands related to buffers

19) pd_sql_object_cache_size = *SQL-object-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((22-256000))**
- 64-bit mode: **((22-2000000))**

Specifies in kilobytes the size of the buffer area (shared memory) in which SQL objects are to be placed.

Specification guidelines

- SQL objects are saved in a buffer until the user's transaction has terminated. The buffer must be large enough to store the SQL objects of all transactions that will be executed concurrently.
- When the hit rate is low for SQL object buffers, performance might degrade due to the overhead required for SQL parsing processing.
- SQL analysis can be reduced by saving the SQL objects of static SQLs in the buffer after transaction termination (until the buffer runs out of space) and sharing them among multiple users who execute the same UAP. To effectively utilize the buffer, allocate it so that the SQL objects of frequently-used UAPs are resident in the buffer.
- To estimate the buffer size, first determine the buffer size needed for UAP execution from the length of the SQL objects from the SQL statements to be issued by the UAP. Then, compute the buffer size by considering the number of UAPs that will be executed concurrently and the number of concurrently executing users.
- For details about how to estimate the length of the SQL object from a single SQL statement, see *C.2 Formulas for determining size of SQL object buffer (pd_sql_object_cache_size)*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition or in the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition and the system common definition, the following value is assumed:

(*value of pd_max_users* + 3) × 22

Tuning the specified value

For details about tuning the SQL object buffer size, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_sql_object_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

20) pd_table_def_cache_size = *table-definition-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((100-65535))**
- 64-bit mode: **((100-2000000))**

Specifies in kilobytes the size of the buffer (shared memory) for table and sequence generator definition information. This information is used during SQL statement pre-processing. Definition information stored in this buffer is managed by the LRU method.

Advantages

- Table and sequence generator definition information that has been used is retained in memory as long as possible so that it can be used again without an input operation.
- Performance improves when a large number of dynamic SQLs are used.

Specification guidelines

- Specify the total size of the definition information needed for frequently-used tables and sequence generators.
- The table definition information buffer size per sequence generator is 8 kilobytes.
- For details about determining the size of the table definition information buffer per table, see *C.4 Formulas for determining size of table definition information buffer (pd_table_def_cache_size)*. For temporary tables, determine the size in the same manner as for non-partitioned tables.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is the applicable value shown below:

$《\uparrow \sqrt{\textit{value of pd\_max\_users}} \uparrow \times 100 + 3》$

Tuning the specified value

For details about how to tune the size of the table definition information buffer, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_table_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

21) pd_auth_cache_size = *user-privilege-information-buffer-size*

~<unsigned integer>((1-100)) (kilobytes)

Specifies in kilobytes the size of the buffer (shared memory) for user privilege information.

Specification guidelines

- The user privilege information buffer stores CONNECT privilege, DBA privilege, and audit privilege information. If this buffer contains no information, information is obtained from a dictionary table during HiRDB connection, thus lengthening the response time. Therefore, specify a buffer size that can store the information for the users who are always connected.
- Storing the user privilege information of each user requires 68 bytes. Use this information when computing the total buffer size.

Tuning the specified value

For details about how to tune the size of the user privilege information buffer, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `1`.

Effects on individual estimation formulas

If the value of the `pd_auth_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

22) pd_view_def_cache_size = *view-analysis-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((0-32000))**
- 64-bit mode: **((0-2000000))**

Specifies in kilobytes the size of the buffer (shared memory) for view analysis information.

Advantage

View analysis information that has been used is kept in the shared memory and can be used subsequently without an I/O operation.

Specification guideline

The total size of the view analysis information for frequently-used view tables is specified. For details about determining the size of the view analysis information buffer, see *C.3 Formulas for determining size of view analysis information buffers (pd_view_def_cache_size)*.

Tuning the specified value

For details about how to tune the size of the buffer for view analysis information, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the value shown below:

《↑ $\sqrt{(\textit{value of pd\_max\_users} + 3)}$ ↑ × 8》

Effects on individual estimation formulas

If the value of the `pd_view_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

23) pd_type_def_cache_size = *user-defined-type-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((100-65536))**
- 64-bit mode: **((100-2000000))**

This operand is applicable to user-defined types; specification of this operand is recommended when user-defined types will be used.

Specifies in kilobytes the size of the buffer (shared memory) for information on user-defined types. When a user-defined type is used, information on it is stored in this buffer. This information is used during pre-processing of SQL statements.

Advantages

- When user-defined type information is stored in this buffer, it is not necessary to access the dictionary table when the same user-defined type is used subsequently, thus reducing the number of input operations and the CPU usage time.
- Specification of this operand improves performance when many dynamic SQLs are used.

Specification guidelines

Specify the total size of the definition information for frequently-used user-defined types that are defined in tables. The following formula can be used to determine the definition information size for one user-defined type:

$\uparrow \{((0.3 + 0.2 \times a + 0.1 \times b) + 3) \div 4\} \uparrow \times 4$ (kilobytes)

a: Number of user-defined type attributes

b: Number of subtypes that have inherited a supertype

Tuning the specified value

For details about how to tune the size of the buffer for user-defined type information, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Effects on individual estimation formulas

If the value of the `pd_type_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

24) pd_routine_def_cache_size = *routine-definition-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((0, 20-65536))**
- 64-bit mode: **((0, 20-2000000))**

Specifies in kilobytes the size of the buffer (shared memory) for storing the following types of definition information (this information is used during pre-processing of SQL statements):

- Plug-in facility definition information
- System definition scalar facility definition information
- Routine definition information

Advantages

When these types of definition information are stored in this buffer, it is not necessary to access the dictionary table when the same information is used subsequently, thus reducing the number of I/O operations and CPU usage time.

Application criteria

Specify this operand when a large number of the following types of SQL statements are used:

- SQL statements that use a plug-in
- SQL statements that use the system definition scalar facility
- SQL statements that use a routine

Specification guidelines

For details about how to determine the value for this operand, see *C.6 Formulas for determining size of routine definition information buffer (pd_routine_def_cache_size)*.

Tuning the specified value

For details about how to tune the size of the routine definition information buffer, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `100`.

Note

If the value specified for this operand is smaller than the total of the sizes of the definition information for all plug-ins, the definition information on plug-in facilities will not be allocated in the buffer.

Effects on individual estimation formulas

If the value of the `pd_routine_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

25) pd_registry_cache_size = *registry-information-buffer-size*

~<unsigned integer>((0-65536)) (kilobytes)

This operand is related to plug-ins. If you use a plug-in that uses registry information, we recommend that you use this operand. We also recommend that you use this operand if you use HiRDB Text Search Plug-in.

Specify the size of the buffer (shared memory) for storing registry information (in kilobytes). When registry information is used, it is stored in the buffer. Registry information is used during the execution of an SQL statement.

Advantages

- Once registry information is stored in this buffer, it is not necessary to access the registry when the same information is used subsequently, thus reducing the number of I/O operations and CPU usage time.
- Specifying this operand can improve performance when a plug-in that makes frequent use of registry information is used.

Specification guidelines

Use the following formula to determine the registry information buffer size:

$\uparrow (0.3 + a) \uparrow \times b$ (kilobytes)

a: Average registry key length (kilobytes)

The average registry key length can be determined with the following SQL statement:

```
SELECT AVG(KEY_LENGTH) FROM MASTER.SQL_REGISTRY
```

Because the result of this SQL statement is output in bytes, convert it to kilobytes.

b: Number of registry keys registered

The number of registry keys registered can be determined with the following SQL:

```
SELECT COUNT(*)FROM MASTER.SQL_REGISTRY
```

Tuning the specified value

For details about how to tune the size of the buffer for registry information, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `10`.

Effects on individual estimation formulas

If the value of the `pd_registry_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

5.2.8 Operands related to shared memory

26) pd_sds_shmpool_size = *single-server-shared-memory-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-200000))**
- 64-bit mode: **((1-4000000000))**

Specifies the size of the area (in kilobytes) to be used by a single server as part of the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand. If it is omitted, HiRDB computes a value for it (however, if `v6compatible` is specified for the `pd_sysdef_default_option` operand, `1024` is assumed). HiRDB computes this value based on the values for the related definitions. Therefore, specify the appropriate values for the related definitions. For details about the related definitions, see *Formulas for shared memory used by a single server* in the *HiRDB Version 9 Installation and Design Guide*.

If the values of any of the operands of the variables used in the estimation formula are changed, HiRDB automatically re-calculates this operand value.

In the formula, for the variables *Number of global buffer pools for index* and *Total number of global buffers (number of pdbuffer operands)*, 500 is assumed in the 32-bit mode and 1,000 is assumed in the 64-bit mode.

Tuning the specified value

If any of the following messages is output, increase the specification value of this operand.

- `KFPA20003-E`
- `KFPD00005-E`
- `KFPD00012-E`
- `KFPD00021-E`
- `KFPH20003-E`

Notes

If this operand is omitted and the appropriate values are not specified in the related definitions, the value obtained by HiRDB might be too large, resulting in the allocation of an unnecessarily large amount of shared memory, or too small, resulting in one or more of the following problems:

- HiRDB does not start.
- A UAP or utility does not execute.

If the value specified for this operand is unnecessarily large or unnecessarily small, the same problems also occur.

Effects on individual estimation formulas

If the value of the `pd_sds_shmpool_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a single server*

5.2.9 Operands related to RPC trace information

27) pd_rpc_trace = Y | N

Specifies whether RPC trace information is to be collected. HiRDB maintenance information is output in the RPC trace information.

Normally, omit this operand.

`Y`: Collect RPC trace information.

`N`: Do not collect RPC trace information.

Note

Specifying `Y` for this operand degrades communication performance.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `N` is assumed.

28) pd_rpc_trace_name = "*name-of-RPC-trace-collection-file*"

~<path name of up to 254 characters>

Specifies an `absolute` path name for the file name for the RPC trace files. Three RPC trace files are created, with 1, 2, and *l* suffixed to the specified file name.

Note

Files with a maximum size of `pd_rpc_trace_size` *value* × 2 are created under the directory specified by this operand. Pay attention to the file size.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, the following value is assumed:

`%PDDIR%\spool\rpctr`

29) pd_rpc_trace_size = *RPC-trace-collection-file-size*

~<unsigned integer>((1024-2147483648)) (bytes)

Specifies in bytes the size of the RPC trace files.

The value specified by this operand determines the size of both RPC trace file 1 and RPC trace file 2.

Specification guidelines

An RPC trace collects in a single file in time order the information on the communications among all processes. If the volume of this information exceeds the specified value, trace data output is switched to the other file, which means that older trace information will be overwritten. When this happens, there might not be enough trace information available, making troubleshooting difficult. For this reason, specify at least `1000000` for this operand.

Note

The value specified by this operand does not apply to RPC trace file *l*, because its size is fixed at 0 bytes.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `4096` is assumed.

5.2.10 Operands related to troubleshooting information

30) pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*

~<unsigned integer>((126-16383))

A HiRDB process records the history of the executed functions and macros inside the process private memory. This history is called a *module trace*. This operand specifies the number of module trace records. The content of this history is loaded into the `core` file and is output when a process error occurs.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `126`.

Note

Process private memory of the following size is allocated to each process:

In the 32-bit mode: 64 + 48 × value of `pd_module_trace_max` *operand* (bytes)

In the 64-bit mode: 64 + 64 × value of `pd_module_trace_max` *operand* (bytes)

31) pd_module_trace_timer_level = 0 | 10 | 20

Specifies how to acquire the time to be output in module traces. The following table explains the meaning of the value specified for this operand.

Specified value	Time acquisition method
`0`	Time is output in seconds at every module trace output location.
`10`	Time is output in microseconds only at performance-critical module trace output locations, such as those before and after input/output processing, and time is output in seconds at other locations.
`20`	Time is output in microseconds at every module trace output location.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `0`.

Note

If you specify a value other than `0` for this operand, a function for acquiring time in microseconds is issued, and as a result, system performance might decline.

32) pd_pth_trace_max = *maximum-number-of-stored-communication-traces*

~<unsigned integer>((1024-8388608))

Specifies the maximum number of communication trace records to be used as troubleshooting information.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reasons such as performance checking, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definitions, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `1024`.

Notes

Increasing the value of this operand increases the amount of process private memory secured by HiRDB processes.

Process private memory for communication traces is calculated based on this operand's value rounded up to the power of two. For details about memory requirements, see *Calculation of required memory* in the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_pth_trace_max` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*

5.2.11 Operands related to global buffers

33) pd_max_add_dbbuff_no = ***maximum-global-buffers-count-for-dynamic-addition***

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, this operand specifies the maximum number of global buffers (per server) that can be added dynamically by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

- Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify a sufficient value based on that value.
- Determine the operand's value in such a manner that the following condition is satisfied:
  Value of `pd_max_add_dbbuff_no` $\leq$ 2,000,000 - number of global buffers allocated per server during HiRDB startup

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, one of the following values is assumed:

Condition		Default value
32-bit mode	$a \geq 500$	`256`
	$a < 500$	500 - *a*
64-bit mode	$a \geq 1,000$	`256`
	$a < 1,000$	1,000 - *a*

a: Number of global buffers allocated per server during HiRDB startup

Notes

Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the value of S* under *Determining the size of status files*
- *Formula 2*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for shared memory used by a single server*

34) pd_max_add_dbbuff_shm_no = ***maximum-shared-memory-segments-count-for-dynamic-addition***

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, specifies the maximum number of shared memory segments (per server) that can be allocated when dynamic addition is performed by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify an appropriate value.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, one of the following values is assumed:

Condition		Default value
`pd_max_add_dbbuff_no` operand is omitted,	32-bit mode	500 + *A*
	64-bit mode	1,000 + *A*
`pd_max_add_dbbuff_no` operand is specified,	32-bit mode	↓ *value of pd_max_add_dbbuff_no* × 1.5 + *A* ↓ (If the value is 32,752 or greater, `32752` is set)
	64-bit mode	

A: Remaining number of shared memory segments that can be allocated during HiRDB startup. This value can be calculated using the following formula:

A = *a* - *b*

Assign the following values to *a* and *b*:

a = *value of pd_max_dbbuff_shm_no (in 64-bit mode*, 16)

b = *number of shared memory segments allocated to each server during HiRDB startup*

You can obtain information about the shared memory segments by using the `pdls -d mem` command or an OS command.

Notes

- If the following condition is satisfied, the value of the `pd_max_add_dbbuff_no` operand is assumed in this operand:

  Value of `pd_max_add_dbbuff_shm_no` < value of `pd_max_add_dbbuff_no`

  The value of the `pd_max_add_dbbuff_no` operand is also assumed when the default value satisfies the above condition.

- Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.

- If the size of a shared memory segment to be added exceeds the `SHMMAX` operand value, shared memory is divided into multiple segments based on the `SHMMAX` operand value as the maximum value. Either increase the value of the `SHMMAX` operand based on the size of the shared memory segment to be added or increase the value of the `pd_max_add_dbbuff_shm_no` operand so that no shortage occurs when the shared memory is segmented.

- If the facility for dynamically changing global buffers is used, the total number of shared memory segments that are allocated for the global buffers can be calculated using the following formula:

  pd_max_dbbuff_shm_no value + *pd_max_add_dbbuff_shm_no value*

  Therefore, if more shared memory segments were allocated when HiRDB started than the number specified for `pd_max_dbbuff_shm_no`, the number of shared memory segments that are actually allocated during dynamic change is the value of `pd_max_add_dbbuff_shm_no` minus the excess number of shared memory segments.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`

- `pd_max_add_dbbuff_no`
- `pd_max_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_shm_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*
- *Determining the value of S*
- *HiRDB/Single Server* under *Determining Environment Variables Related to the Number of Resources*

5.2.12 Operands related to temporary tables

35) pd_max_temporary_object_no = *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*

~<unsigned integer> ((0-131072))

Specifies the maximum number of temporary tables and temporary table indexes that can be used at any one time for each server.

Specification guidelines

Use the formula shown below to determine the value of this operand. For a HiRDB/Parallel Server, obtain the value for each back-end server.

Maximum value of (*the number of transaction-specific temporary tables*
+ *total number of temporary table indexes for those temporary tables*) × *number of activities*#
+ (*number of SQL session-specific temporary tables used in the SQL session*
+ *total number of temporary table indexes for those temporary tables*) × *number of connected users*

#

number of activities:

((*value of pd_max_users* + 3) × 2 + 1) + α

α : If the value specified for `pd_max_users` is `60` or less, 5; if it is `61` or greater, 0.

For a HiRDB/Parallel Server, this operand value is applied to each back-end server. Therefore, as a guideline, specify the largest value used among all back-end servers in this operand.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition or in the system common definition is used, in that order. If this same operand is also omitted from the server common definition and the system common definition, `0` is assumed.

Notes

The value of this operand affects the size of shared memory used by HiRDB; therefore, do not specify a value that is greater than the value described in *Specification guidelines*. If the specified value is greater than the value described in *Specification guidelines*, HiRDB might not be able to start due to a shortage of shared memory.

Relationship to other operands

This operand is related to the `pd_max_tmp_table_rdarea_no` operand.

Effects on individual estimation formulas

If the value of the `pd_max_temporary_object_no` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 8* under *Formulas for shared memory used by a single server*

5.2.13 Operands related to security

36) pd_audit_def_buffer_size = *security-audit-information-buffer-length*

~<unsigned integer>((1-2000000)) (kilobytes)

Specifies (in kilobytes) the buffer size (shared memory) used for storing information for the security audit facility.

Specification guidelines

Use the following formula to determine the security audit information buffer length:

$\uparrow 0.3 + a \times 0.25 \uparrow$ (kilobytes)

a: Number of objects for the narrowing condition that was specified in the audit trail of the security audit facility

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the buffer size shown below is acquired during HiRDB startup. If this amount of memory cannot be allocated, HiRDB will start, but the security audit information buffer is not created. In this case, the `KFPD00032-W` message is issued.

$\uparrow 0.3 + \max\{(a + 100), (a \times 1.2)\} \times 0.25 \uparrow$ (kilobytes)

a: Number of objects for the filtering condition that was specified in the audit trail of the security audit facility

Notes

If the amount of memory required by this operand's specification cannot be allocated, HiRDB will not start.

Effects on individual estimation formulas

If the value of the `pd_audit_def_buffer_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Single Server*

5.2.14 Operands related to delayed batch creation of plug-in index

37) pd_plugin_ixmk_dir = "*index-information-file-creation-directory-name*" or "*index-information-file-creation-HiRDB-file-system-area-name*"

~<path name>

Specifies the name of the directory under which the index information file for delayed batch creation of a plug-in index is to be created. Specify a HiRDB file system area name in order to create the index information file in a HiRDB file system area. An absolute path name must be used for the directory name or HiRDB file system area name.

For details about delayed batch creation of a plug-in index, see the *HiRDB Version 9 System Operation Guide*.

Notes

- The directory (or HiRDB file system area) specified here must have been created in advance. If a nonexistent directory (or HiRDB file system area) is specified, an error will result during execution of a UAP that specifies delayed batch creation of plug-in indexes (UAP that is executed in an environment in which `PDPLGIXMK = YES` is specified in the client environment definition).
- Once the UAP has executed, the value specified for this operand must not be changed before delayed batch creation of plug-in indexes is performed by the database reorganization utility. If it is changed, a plug-in index delayed batch creation cannot be performed.

5.2.15 Operands related to Java

Java operands are specified when a Java stored procedure or Java stored function is used. For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide.*

38) pd_java_stdout_file = *"Java-virtual-machine-standard-output-and-standard-error-output-destination-file"*

~<path name>

Specifies as an absolute path name the file to which the standard output and standard error output are to be output in a Java virtual machine.

Specification guideline

Because the size of the file specified by this operand is extremely large, this operand is not normally specified. It is recommended that this operand be specified during debugging of a Java stored procedure or Java stored function. There is no limit to the size of the file that can be specified by this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definitions, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the Java Virtual Machine standard output and standard error output are ignored.

Note

If there are simultaneous writing attempts from multiple processes, their output contents cannot be guaranteed.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

39) pd_java_castoff = Y | N

Specifies whether to use the following events as triggers for shutting down the process at the server (single server, front-end server, dictionary server, or back-end server) that started the Java Virtual Machine:

No.	Server type	Process name	Trigger that ends process
1	Single server	`pdsds`	UAP is disconnected
2	Front-end server	`pdfes`	UAP is disconnected
3	Dictionary server	`pddic`	Transaction is completed or UAP is disconnected
4	Back-end server	`pdbes`	Transaction is completed or UAP is disconnected

`Y`: Shut down server process when trigger occurs.

`N`: Do not shut down server process when trigger occurs.

Specification guidelines

Normally, this operand is not specified. However, if you encounter the problems described below, we recommend that you specify `Y` in this operand:

- Use of the Java Virtual Machine causes the amount of memory usage to increase to the point where the available system memory becomes nearly exhausted.
- SQL code that includes numerous search conditions is executed, and even though the connection does not use the Java Virtual Machine, the maximum stack size set by the Java Virtual Machine on another connection prevents the stack from expanding, causing the server process to be aborted by a segmentation error.

For details about the Java Virtual Machine facility, see the Java Virtual Machine documentation.

Notes

On systems that run Java stored routines frequently, specifying Y in this operand will generate overhead for server process restarts and Java Virtual Machine startups.

Relationship to other operands

This operand is related to the `pd_process_count` operand.

5.2.16 Operands related to system log files

40) pd_log_dual = Y | N

Specifies whether dual system log files are to be used.

`Y`: Use dual system log files.

`N`: Do not use dual system log files.

Advantages

When dual system log files are used, HiRDB collects the same system log information in both files, which are called File A and File B. If a failure occurs in one of the files while the collected system log file is being loaded, the file can be loaded from the other file, resulting in higher system reliability.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

When use of dual system log files is specified, the name of the File B system log file must be specified with the `pdlogadpf` operand.

41) pd_log_remain_space_check = warn | safe

Specifies the processing to be performed by HiRDB when the available space in the system log file falls below the warning level. This is called the facility for monitoring the free area for the system log file. For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

`warn`:

When the available space in the system log file falls below the warning level, the `KFPS01162-W` message is output.

`safe`:

When the available space in the system log file falls below the warning level, scheduling of new transactions is suppressed, all transactions inside the server are forcibly terminated, and the `KFPS01160-E` message is output.

Specification guideline

We recommend that you specify `safe` because it can reduce the probability of abnormal termination of units due to system log file space shortage. However, when `safe` is specified, all transactions inside the server are forcibly terminated when a shortage occurs in the available space in the system log file. Therefore, the design of the system log file requires more accuracy. For details about system log file design, see the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `warn`.

42) pd_log_auto_unload_path = "*unload-log-file-output-directory*"[,"*unload-log-file-output-directory*"]...
~<path name>((1-136 characters))

Specifies as absolute path names the unload file output directories when the automatic log unloading facility is to be used for the system log. A HiRDB file system area name must be specified to create the unload log file in a HiRDB file system area. *The directories or HiRDB file system areas specified for this operand must be created before HiRDB is started.*

For details about the automatic log unloading facility, see the *HiRDB Version 9 System Operation Guide*.

Specification guidelines

It is important to check the available disk space before specifying a directory, so as to ensure that the created unload log file does not cause a disk space shortage.

If an unload log file cannot be created in the specified directory because of a disk space shortage, the automatic log unloading facility stops. If this is a possibility, creation of multiple directories is recommended.

Note, however, that the database recovery operation of selecting the unload files needed for recovery is simplified somewhat when only one directory is used.

Also keep in mind the following when multiple directories are created:

- It is recommended that directories be specified in different partitions to protect against disk errors.
- If the unload log file cannot be created in a single directory because of a full disk or disk error, create an unload log file under a different directory. HiRDB uses the directories specified by this operand in the order of their specification.

Operand rules

- Up to 128 directories can be specified.
- When multiple directories are specified, the same path name cannot be specified.

Notes

- The automatic log unloading facility cannot be used in the following cases:
  - N is specified in the pd_log_unload_check operand.
- If two or more directories are created for unloaded log files and there is no empty directory when HiRDB starts normally, the automatic log unloading facility stops.
- When a multi-HiRDB is being used, you must create a different directory for each HiRDB. Specifying the same directory for more than one HiRDB will make it impossible to determine which unload log file applies to which HiRDB.

43) pd_log_auto_unload_restart = Y | N

Specifies whether the automatic log unloading facility is to be restarted when the system log files are swapped following the issuance of a message indicating that the automatic log unloading facility stopped due to an error (message KFPS01150-E).

Y: Restart the automatic log unloading facility.

N: Do not restart the automatic log unloading facility.

Condition

The following two conditions must be satisfied:

- Y is specified in the pd_log_unload_check operand or this operand is omitted.
- The pd_log_auto_unload_path operand is specified.

Advantages

- If Y is specified in this operand and a shortage of disk capacity occurs due to an increase in the number of work files, or because the unload processing fails due to a temporary error such as a process creation error, HiRDB automatically restarts the unloading of system logs the next time the system log files are swapped.
- If N is specified in this operand and the unload log files are set to be saved while the automatic log unloading facility is stopped due to an error, you can prevent the unload log files from being saved before the pdlogatul -t command is executed.

Specification guidelines

Normally, specify Y or omit this operand.

Specify N if you have already set HiRDB to monitor the automatic log unloading facility termination message (KFPS01150-E message) and restart the facility with the pdlogatul -b command.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, Y is assumed.

44) pd_log_singleoperation = Y | N

This operand is applicable when dual system log files are used; it need not be specified when dual system log files are not used.

Specifies whether the single-operation mode is to be used for the system log files. Even if an error occurs in a system log file, making no dual system log files available, HiRDB can continue processing using the remaining single normal system log file without being abnormally terminated. This is called *single operation of the system log files*.

The mode in which continuation of processing is permitted only with both system log files available (normal operation mode) is called *double operation of the system log files*.

Y: Use single operation of the system log files.

N: Do not use single operation of the system log files. Both system log files must always be used.

Condition

This operand is valid only when pd_log_dual = Y is specified.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

45) pd_log_rerun_reserved_file_open = Y | N

Specifies whether a system log file is to be opened automatically.

If no system log file that can be overwritten is available during a HiRDB restart, HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of system log file*.

A reserved file is used in the following cases:

- Between the time of a restart and the time when the first synchronization point dump is collected
- When none of the opened file groups can be overwritten

`Y`: Open a system log file automatically (open and use a reserved file).

`N`: Do not open a system log file automatically (do not use a reserved file).

Advantages

When `Y` is specified, the unit can be restarted as long as a reserved file is available, even though no file that can be swapped in is available during the unit restart.

However, if a file that is in unload wait status is available, the unit is stopped; once that file has been unloaded, the unit can be restarted.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

46) pd_log_rerun_swap = Y | N

Specifies whether the system log files are to be swapped during a HiRDB restart.

`Y`: Swap the system log files.

`N`: Do not swap the system log files.

Advantage

When `Y` is specified, there can be a physical separation between the system log files used before and after a restart. Therefore, the system log file that was being used before the restart can be reused during server operation.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

47) pd_log_swap_timeout = *wait-time-for-completion-of-system-log-file-swapping*
~<unsigned integer>((1-32580)) (seconds)

Specifies in seconds the wait time during which swapping of system log files must be completed. If a swap of system log files is not completed within the specified amount of time, the unit terminates abnormally.

Specification guidelines

Normally, there is no need to specify this operand. If it takes a long time to swap system log files for a reason such as poor machine performance, you can specify this operand with a value that is greater than the default value. To detect errors or delays quickly during system log file swapping so that the unit can be terminated (such as because of a disk failure), reduce the value of this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `180`.

48) pd_log_unload_check = Y | N

Specifies whether HiRDB is to check the unload status of system log files.

`Y`:

Check the unload status (normal operation).

`N`:

Do not check the unload status. A system log file is placed in swappable status, regardless of its unload status, when both of the following conditions are satisfied:

- It is in overwritable status
- It is in extraction completed status (HiRDB Datareplicator)

In this case, the system log file operation method is to release checking of unload status. For details about this operation method, see the *HiRDB Version 9 System Operation Guide*.

Advantages

Specifying `N` provides the following advantages:

- Operations are simplified because the system log file unload operation is eliminated.
- It is not necessary to provide files for storing unload files.

Specification guideline

Specify `N` if the system log file will not be needed for database recovery (in other words, if recovery from a backup collection point will be sufficient).

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `Y`.

Notes

The following points apply when `N` is specified:

- Database can be recovered only if backups have been made.
- If this option is specified when the system log file is required for database recovery, it will not be possible to recover the database.

49) pd_log_max_data_size = *log-input/output-buffer-size*

~<unsigned integer>((32000-523000)) (bytes)

Specifies in bytes the size of the buffer to be used for system log input/output operations.

Specification guideline

Change the specification value according to the following tuning method.

Tuning the specified value

A value (other than the default value) might need to be specified for this operand after the following types of statistics analysis utility statistical information related to system operation have been checked:

- Number of buffer sectors waiting for input/output (`# OF BUFFER FOR WAIT I/O`)

  If the average number of buffer sectors waiting for input/output significantly exceeds 100, increase the value for this operand so that the average approaches 100.
- Number of waits caused by lack of a current buffer (`# OF WAIT THREAD`)

  If the number of waits caused by lack of a current buffer is not 0, increase the value for this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `400000`. (However, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `32000`.)

Notes

The specification of this operand affects the response and throughput of SQL code executed by a transaction. When a small value is specified, writing to system log files occurs frequently, which might result in deterioration of performance.

Relationship to other operands

Use this operand and the `pd_log_write_buff_count` operand to determine the log I/O buffer size.

50) pd_log_write_buff_count = *log-output-buffer-sectors-count*

~<unsigned integer>((3-65000))

Specifies the number of buffer sectors to be used for system log output.

Tuning the specified value

Specify `10` (the default) for this operand initially. Thereafter, use a statistics analysis utility to obtain statistical information related to system operation to check the number of waits caused by a shortage of current buffer space (# OF WAIT THREAD). If the number of waits is high, specify a larger value to improve throughput.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `10`. (However, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `3`.)

Notes

If a small value is specified in this operand and the number of transactions is high, multiple transactions might be waiting for system log output, lowering system performance.

Relationship to other operands

Use this operand and the `pd_log_max_data_size` operand to determine the number of log output buffer sectors.

51) pd_log_rec_leng = *system-log-file-record-length*

~<unsigned integer>((1024, 2048, 4096)) (bytes)

Specifies the record length for the system log files; the specifiable values are `1024`, `2048`, and `4096`.

Specify the record length specified in the `-l` option of the `pdloginit` command for this operand.

Specification guidelines

Specify the record length based on the guidelines for designing the record length of system log files. For details about the guidelines for designing the record length of system log files, see *Record length of a system log file* in the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `4096`.

Notes

- If a value that is different from the record length specified by the `-l` option of the `pdloginit` command is specified for this operand, system log files cannot be opened.
- For details about how to modify the system log file record length, see the *HiRDB Version 9 System Operation Guide*.

52) pd_log_rollback_buff_count = *rollback-log-input-buffer-sector-count*

~ <unsigned integer>((0-256))

Specifies the number of buffer sectors to be used for system log input during rollback processing. When `0` is specified in this operand, HiRDB determines the number of rollback log input buffer sectors.

Specification guidelines

- We recommend that you specify `0` in this operand, in which case HiRDB will calculate an appropriate value automatically.
- If the specified value is too small, concurrent execution of rollbacks might be slowed. If the specified value is too large, the unit controller might use more shared memory than is necessary.

Tuning the specified value

Specify `0` in this operand. If specifying `0` leads to memory shortages, do not specify this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the following value is assumed:

Number of servers in the unit $\times$ 2

Effects on individual estimation formulas

If the value of the `pd_log_rollback_buff_count` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Single Server*

53) pd_log_auto_expand_size = *extension-amount-per-system-log-file-extension-trigger*[,*extension-limit*]

~<unsigned integer>((0-104857600))<<0,0>>(records)

Specify this operand when the system log file automatic extension facility is used.

Specifies the number of records to be added to the system log file for each extension trigger and an upper limit for extending file size.

If the amount to be added per trigger is omitted or `0` is specified, the system log file will not be extended automatically. If the extension limit is omitted or `0` is specified, the system log file might be extended until either the disk on which the file system area is located becomes full or the system log file capacity reaches its upper limit. When a value is specified for the amount to be added per trigger that is larger than the extension limit, the file will be extended up to the extension limit.

For details about the system log file automatic extension facility, see the *HiRDB Version 9 System Operation Guide*.

Condition

`Y` must be specified in the `pd_large_file_use` operand or specification of this operand must be omitted.

Specification guidelines

- Specify in this operand the amount to be added per trigger based on the number of records specified when the system log file was created with the `-n` option of the `pdloginit` command. Determine 10 percent of the average number of records for all system log files and specify that value.
- Normally, an extension limit is omitted.

Tuning the specified value

If the system log output volume exceeds the expanded size after automatic extension, the system log file could become full, resulting in a unitdown. In such a case, increase the specified value (number of records to be added per trigger). If extension processing requires so much time that it affects transaction performance, specify a smaller value.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed.

5.2.17 Operands related to synchronization point dump files

54) pd_spd_dual = Y | N

Specifies whether to use dual synchronization point dump files.

`Y`: Uses dual synchronization point dump files.

`N`: Does not use dual synchronization point dump files.

Advantage

When dual synchronization point dump files are used, HiRDB collects the same synchronization point dump in both dump files A and B. Even when an error occurs in one of the files when the collected synchronization point dump is being loaded, the synchronization point dump can be loaded from the other file, thus improving system reliability.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

To use dual synchronization point dump files, specify the name of synchronization point dump file B in the `pdlogadpf` operand.

55) pd_spd_assurance_msg = Y | N

Specifies whether the `KFPS02183-I` message is to be output when a synchronization point dump is completed.

`Y`: Output the message.

`N`: Do not output the message.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `Y`.

56) pd_spd_assurance_count = *number-of-guaranteed-valid-generations*

~<unsigned integer>((1-2))

Specifies the range of system log files to be saved during HiRDB operation as a protection against events such as a synchronization point dump file input error during system recovery. What is specified here is a number of synchronization point dump file generations; the specified value is referred to as the *number of guaranteed-valid generations*. The number of past generations of synchronization point dump files specified here are overwrite disabled.

Advantage

When `2` is specified as the number of guaranteed-valid generations and an error occurs in the most recent synchronization point dump file generation, the system can be recovered using the preceding synchronization point dump file generation, resulting in higher reliability.

Specification guidelines

- To improve reliability, specify `2` for the number of guaranteed valid generations. However, the number of overwrite disabled synchronization point dump files increases (to two).
- If reliability has been improved with the use of dual synchronization point dump files, we recommend that you omit this operand or specify `1` for it.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `1`.

Notes

- The minimum number of synchronization point dump files needed is *number-of-guaranteed-valid-generations* + 1.
- Specifying `2` increases the number of overwrite disabled synchronization point dump files. Moreover, the system log files that correspond to the overwrite disabled synchronization point dump files are also overwrite disabled. Consequently, specifying `2` for the number of guaranteed valid generations increases the number of overwrite disabled system log files. As a result, a shortage might occur in the number of system log files that can be swapped in. To prevent this, it might be necessary to re-evaluate the system log file capacity.

57) pd_spd_reduced_mode = *reduced-mode-operation-option*

~<unsigned integer>((0-2))

Specifies whether the reduced mode operation for synchronization point dump files is to be used.

Reduced mode operation is a facility for continuing processing if at least two synchronization point dump files can be used, even if an event such as a file error during HiRDB operation or restart reduces the number of synchronization point dump files to or below the number of required files (*number of guaranteed valid generations*# + 1).

#: Value specified for the `pd_spd_assurance_count` operand.

`0`: Do not use the reduced mode operation.

`1`: Use the reduced mode operation.

`2`: Use the reduced mode operation and issue a warning message whenever a synchronization point dump is being acquired during the reduced mode operation.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

58) pd_spd_reserved_file_auto_open = Y | N

Specifies whether a synchronization point dump file is to be opened automatically. When `Y` is specified and an error in a synchronization point dump file reduces the number of synchronization point dump files to the number of files necessary for operation (*number of guaranteed valid generations*# + 1), HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of synchronization point dump file*.

#: Value specified for the `pd_spd_assurance_count` operand.

`Y`:

Open a synchronization point dump file automatically. A reserved file is opened when the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

`N`:

Do not open a synchronization point dump file. No reserved file is opened, even though the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

This operand has a higher priority than the `pd_spd_reduced_mode` operand.

59) pd_spd_max_data_size = *synchronization-point-dump-file-buffer-size*

~<unsigned integer>((32000-4000000)) (bytes)

Specifies in bytes the size of the buffer (shared memory) to be used for synchronization point dump file input/output operations.

The value specified here affects the number of synchronization point dump file input/output operations.

Specification guidelines

- Normally, this operand need not be specified.
- When the value of this operand is increased, the number of synchronization point dump file input/output operations is reduced.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `32768`.

60) pd_log_sdinterval = *system-log-output-volume*[,*interval*]

Specifies the collection interval for synchronization point dumps. This operand can be specified based on the following information:

- Volume of system log information
- Amount of time that has elapsed since collection of the previous synchronization point dump

***system-log-output-volume*: ~<unsigned integer>((100-100000)) (number of log blocks)**

Specifies an interval between synchronization point dumps in terms of the number of blocks of log information. A synchronization point dump is collected each time system log information equivalent to the number of log blocks specified here has been output.

***interval*: ~<unsigned integer>((0 or 10-1440)) (minutes)**

Specifies a synchronization point dump collection interval in terms of the number of minutes between synchronization point dumps.

- If `0` is specified for the interval, HiRDB does not use a time interval for collecting synchronization point dumps.
- If no transactions execute during the time interval since the previous synchronization point dump was collected, no synchronization point dump is collected, even if the interval specified here has elapsed.

Specification guidelines

- This operand need not be specified if no specific amount of time is specified for restarting HiRDB.
- The value specified for this operand affects the amount of time required to restart HiRDB.

  Specifying a small value for this operand reduces the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps increases, online performance might deteriorate in some cases.

  Conversely, specifying a large value for this operand increases the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps decreases, online performance might improve in some cases.

Tuning the specified value

The synchronization point dump collection intervals can be checked with the statistics analysis utility; the relevant information is shown under `SYNC POINT GET INTERVAL` in the statistical information related to system operation. Use the average of the `SYNC POINT GET INTERVAL` values. If the synchronization point dump collection interval is determined to be too long, decrease the specification value; conversely, if it is determined to be too short, increase the specification value.

Operand default

If this operand is omitted, the specification for the same operand in the server common definition is takes effect. If the same operand is also omitted in the server common definition, the following values are assumed:

- *system-log-output-volume*: `5000`
- *interval*: `60`

However, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default for system log output volume is `1000`.

Note

- The synchronization point dump collection interval is determined based on the volume of system log information that is output. Therefore, committing data from memory to a database takes a long time during intervals that have few updating transactions. If an error occurs at such a time, it will take longer to recover the transactions that were generated during that period. If this is a possibility, also use the `interval` value to set the synchronization point dump collection interval.
- When a synchronization point dump is collected, an update page will be output from the global buffer, increasing the load on the CPU, I/O processing, and lock time. For this reason, reducing the dump interval can delay processing.

5.2.18 Operands related to server status files

61) pd_sts_file_name_1 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

:

pd_sts_file_name_7 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

Define server status files. Although the `pd_sts_file_name_2` to `7` operands can be omitted, the `pd_sts_file_name_1` operand cannot be omitted.

"*logical-file-name*" ~<identifier>((1-8 characters))

Specifies the logical file name of a status file for the single server.

When a command that manipulates the status file is to be executed, the logical file name defined here must be specified.

"*file-a-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File A status file as an absolute path name.

"*file-b-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File B status file as an absolute path name.

Specification guidelines

- The files specified as File A and File B must be status files created with the `pdstsinit` command. If a file that has not been created with the `pdstsinit` command is specified, a virtual status file is created.
- If an error occurs in a status file, HiRDB swaps the status files. If no spare file is available for swapping, the unit terminates abnormally. For this reason, defining a large number of system files improves system reliability, but at the expense of increasing the amount of disk space that is required.
- The status files specified for File A and File B must have the same record length and capacity.

Operand rules

- Up to seven instances of this operand can be specified.
- A status file has a dual structure consisting of File A and File B; both must be specified.
- Environment variables cannot be used for the absolute path names of the File A and File B file names.

- The same names cannot be specified for the logical file name, the File A file name, and the File B file name.
- HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sts01`, `C:\hirdb\sysfile` is not case sensitive, but `sts01` is case sensitive.

Notes

- When HiRDB is started normally, the primary file (the file that was the primary file when HiRDB was terminated) is inherited. However, if there is no current file that can be inherited, for example because all status files have been initialized, the first status file that was specified among those specified by the `pd_sts_file_name_1` to `7` operands becomes the current file, and the remaining files become spare files if they can be opened. Any files that cannot be opened become reserved files.
- When HiRDB is restarted, the primary file (the file that was the primary file when HiRDB was terminated) is inherited.

Using a virtual status file

Specifying a virtual status file makes it possible to add a new status file while HiRDB is running.

For example, if the number of spare files is reduced because of errors in status files, virtual status files can be converted into spare files. The procedure for converting virtual status files into spare files follows.

Procedure

1. Use the `pdstsinit` command to create a status file in a HiRDB file system area for system files.
2. Use the `pdstsopen` command to open the status file.

These operations can be performed while HiRDB is running; there is no need to stop HiRDB.

- **Advantages and disadvantages**

  Defining a virtual status file reduces the amount of space required in the HiRDB file system area. However, a virtual status file cannot be added as a spare file if the HiRDB file system area for system files does not have sufficient space (sufficient space for adding the file), thus reducing system reliability.

  When virtual status files are not defined, the amount of space required in the HiRDB file system area is increased. However, because a swap destination is guaranteed when a file error occurs, system reliability is increased.

- **Notes**

  When virtual status files are defined, HiRDB determines that a status file error has occurred when the unit is started. For this reason, HiRDB cannot start if `stop` (default value) is specified for the `pd_sts_initial_error` operand. When virtual status files are defined, specify `continue` or `excontinue` for the `pd_sts_initial_error` operand. It is also necessary before starting HiRDB to specify the current file in the `pd_sts_last_active_file` operand.

5.2.19 Operands related to server status files (when an error occurs)

For details about the measures to be taken when an error occurs in a status file, see the *HiRDB Version 9 System Operation Guide*.

62) pd_sts_initial_error = stop | continue | excontinue

When a server (single server, front-end server, dictionary server, or back-end server) starts, HiRDB performs a process of identifying the current server status file. This operand specifies the action that HiRDB takes when any of the following errors are detected during this identification process.

- No real server status file is found.
- An error is detected in the server status file.

The current file identification process is applied to the server status files specified by the `pd_sts_file_name_1` to `7` operands.

`stop`:

When an error is detected in a server status file during the current file identification process, the startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started. In this case, first take a corrective action for the status file in which the error was detected, and then start HiRDB.

`continue` or `excontinue`:

Even when an error is detected in the server status file during the current file identification process, the startup of the server is continued if the current file is normal. However, the startup might be stopped depending on the value specified for the `pd_sts_singleoperation` operand (whether operation continues with a single status file). The following table shows the relationship to the `pd_sts_singleoperation` operand.

- **Relationship to the pd_sts_singleoperation operand**

pd_sts_singleoperation operand value	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
`continue`	When an error is detected in a server status file, HiRDB cannot identify the current file, and thus startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	The HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.
`stop` (default value)	When an error is detected in a server status file, HiRDB identifies the current file and startup of the server is continued. However, if the primary and secondary files satisfy any of the conditions listed in the table below (cases in which HiRDB cannot identify the current file), startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	If HiRDB cannot identify the current file, the HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.

- **When HiRDB cannot identify the current file**

pd_sts_initial_error operand value	File A status	File B status
`continue`	Error shutdown	Error shutdown
	Error shutdown	Open (initial state)
	Error shutdown	No real file
	Open (initial state)	Error shutdown
	Open (initial state)	No real file
	No real file	Open (initial state)
	No real file	Error shutdown
	No real file	No real file
`excontinue`	Error shutdown	Error shutdown
	Error shutdown	No real file
	No real file	Error shutdown
	No real file	No real file

Therefore, to minimize actions by the HiRDB administrator (to reduce the number of cases in which both the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands must be specified), specify the following:

- `pd_sts_initial_error = excontinue`
- `pd_sts_singleoperation = stop`

Specification guidelines

The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_initial_error operand value	
	stop	continue or excontinue
Processing by HiRDB during server startup	When an error is detected in a server status file, startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	Even when an error is detected in a server status file, the startup of the server is continued if the current file is normal.
Specification guideline	To improve system reliability, specify `stop`.	To simplify the error-handling actions during HiRDB startup, specify `continue` or `excontinue`.
Advantage	Guarantees that all server status files of the server are normal when the server starts. Therefore, if an error occurs in the current file after HiRDB has started, it can be swapped to a spare file.	Even when an error is detected in a server status file during server startup, HiRDB can start with the remaining normal files only. Therefore, HiRDB stop time can be shortened. In this case, because the number of spare files has become small, it is necessary to immediately repair the status files containing errors.
Disadvantage	Possibility increases that an error in a server status file stops the startup of HiRDB.	Because HiRDB might be running with only a small number of spare files, system reliability is low. Depending on the number of spare files available, it might not be possible to swap server status files.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `stop` is assumed.

Notes

- If both current files are abnormal, startup of the server is stopped regardless of the value specified for this operand, and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.
- Before starting HiRDB, do not initialize the current file with the `pdstsinit` command. If the current file is initialized, HiRDB cannot be restarted.
- If `excontinue` is specified for this operand and all generations of the status files are in single-operation mode and open (initial status), HiRDB does not start.

Remarks

The following figure shows the values specified for this operand, the processing performed by HiRDB, and the actions to be taken by the HiRDB administrator.

● Values specified for this operand and the processing performed by HiRDB

pd_sts_initial_error operand value	Status file error	pd_sts_singleoperation operand value	Can HiRDB identify the current file?	Specification of pd_sts_last_active_file	Does the pd_sts_last_active_file operand value match the latest file that can be opened?	Current file error	Specification of the pd_sts_last_active_side operand	Is the file specified for the pd_sts_last_active_side operand usable?	Numbers listed in the Table Processing performed by HiRDB and actions to be taken by HiRDB the administrator
`stop` (default value)	None	—	—	—	—	—	—	—	*[1]*
	Yes	—	—	—	—	—	—	—	*[5]*
`continue`	None	—	—	—	—	—	—	—	*[1]*
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	*[2]*
			Cannot be ID'd.	None	—	—	—	—	*[6]*
				Yes	Matches.	None	—	—	*[3]*
						Yes	None	—	*[9]*
							Yes	Usable	*[4]*
								Not usable	*[7]*
					Does not match.	—	—	—	*[8]*
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	*[6]*
				Yes	Matches.	None	—	—	*[3]*
						Yes	None	—	*[9]*
							Yes	Usable	*[4]*
								Not usable	*[7]*
					Does not match.	—	—	—	*[8]*
`excontinue`	None	—	—	—	—	—	—	—	*[1]*
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	*[2]*
			Cannot be ID'd.	None	—	—	—	—	*[6]*
				Yes	Matches.	None	—	—	*[3]*
						Yes	None	—	*[9]*
							Yes	Usable	*[4]*
								Not usable	*[7]*
					Does not match.	—	—	—	*[8]*
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	*[6]*
				Yes	Matches.	None	—	—	*[3]*
						Yes	None	—	*[9]*
							Yes	Usable	*[4]*
								Not usable	*[7]*
					Does not match.	—	—	—	*[8]*

Legend:

—: Not applicable (the condition does not affect the processing performed by HiRDB)

● Processing performed by HiRDB and the actions to be taken by the HiRDB administrator

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[1]	HiRDB startup processing continues.	None
[2]	HiRDB identifies the latest current file and continues the startup processing.	Make the file that is in the error-shutdown state into a spare file.
[3]	Using the file specified in the `pd_sts_last_active_file` operand as the current status file, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[4]	Using the files specified in the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands as the current status files, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[5]	Because `stop` is specified for the `pd_sts_initial_error` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000010` in message `KFPS01005-E`.

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[6]	Because the current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000016` in message `KFPS01005-E`.
[7]	Because the normal current file identified by HiRDB does not match the file specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000017` in message `KFPS01005-E`.
[8]	Because the current file name identified by HiRDB does not match the file name specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000015` in message `KFPS01005-E`.
[9]	Because the normal current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000018` in message `KFPS01005-E`.

63) pd_sts_singleoperation = stop | continue

Specifies whether processing of server status files continues in the single operation mode. The status file single-operation mode means that processing is to continue using only the normal file (single file) when an error occurs in the status file and a spare file is not available. For details about the status file single-operation mode, see the *HiRDB Version 9 Installation and Design Guide*.

If an error occurs in one of the current files and a spare file is available, the status file is swapped and processing continues regardless of the value specified for this operand (operation in the single-operation mode does not occur).

`stop`:

Do not permit operation in the single-operation mode (the unit is to be terminated abnormally instead). If HiRDB is abnormally terminated, allocate spare files and then restart HiRDB.

`continue`:

Enable operation in the single-operation mode (processing is to continue using the normal side of the current file). When the single-operation mode goes into effect, the message `KFPS01044-I` is output. If an error occurs in the normal file during operation in the single-operation mode or if HiRDB is abnormally terminated while the status file is being updated, HiRDB cannot be restarted. Therefore, when the single-operation mode goes into effect, immediately allocate spare files.

Specification guidelines

- We recommend that you specify `stop` to increase system reliability. We also recommend that you increase the number of spare files to guard against errors in the current files.
- The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_singleoperation operand value	
	stop	continue
Specification guideline	To improve system reliability, specify `stop`.	Specify `continue` if it is important not to stop HiRDB.
Advantage	When an error occurs in one of the current files and a spare file is not available, operation in the single-operation mode does not occur and HiRDB is abnormally terminated. Consequently, the possibility of losing the content of the current files is reduced.	Even when an error occurs in one of the current files and a spare file is not available, processing can be continued. Therefore, the possibility that an error in the status file stops HiRDB is reduced.
Disadvantage	Possibility that an error in the status file stops HiRDB increases. However, increasing the number of spare files can reduce this possibility.	If an error occurs in the normal status file during operation in the single-operation mode or if HiRDB is abnormally terminated during updating of the status file, the content of the current file is lost and HiRDB cannot be restarted.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `stop` is assumed.

Relationship to other operands

The combination of the values specified for the `pd_sts_singleoperation` and `pd_sts_initial_error` operands determines the processing by HiRDB when an error occurs in the status file. Therefore, the values to be specified for these operands must be determined together.

64) pd_sts_last_active_file = "*logical-file-name*"

~<identifier>((1-8 characters))

Specifies the name of the logical file to be used as the current status file at the time of HiRDB startup. HiRDB compares the file name specified in this operand with the file name selected by HiRDB to be the current file. If the file names match, HiRDB is started; otherwise, HiRDB is not started.

Conditions

The following conditions must be satisfied:

- `continue` or `excontinue` is specified in the `pd_sts_initial_error` operand.
- It cannot be determined if the current file selected by the HiRDB system was the most recent current file during the previous session.

Specification guidelines

1. To start HiRDB immediately after initializing all status files:

   From among the operable logical files specified in the `pd_sts_file_name1-7` operands, specify the one that has the smallest number. Forced startup will be used in this case, regardless of the previous termination mode.

2. When both of the current files are normal:

   Specify the name of the current file.# If HiRDB cannot `be` started even though the name of the current file is specified, the current file might have been initialized. In this case, first initialize all status files, and then use the method in step 1 above to start HiRDB (forced startup will be used, regardless of the previous termination mode).

3. When one of the current status files has an error:

   Use the method in step 2 above, with the following operands specified:

   - `pd_syssts_singleoperation = continue`
   - `pd_sts_last_active_side`

4. When both of the current status files have errors:

   Initialize all status files, then execute the method in step 1 above (forced startup will be used, regardless of the previous termination mode).

5. When a virtual status file is specified:

   Specify the name of the current file.#

#: The names of the current files (that were active at the end of the previous operation) can be determined from the following messages:

- `KFPS01001-I`
- `KFPS01010-E`
- `KFPS01011-I`
- `KFPS01063-I`

Of the status files displayed by these messages, the one that is reported in the message that was output most recently is the current file.

65) pd_sts_last_active_side = A | B

Specify this operand if you want to start HiRDB when one of the current files is in an error state. Specify the normal status file for this operand. HiRDB compares the file specified in this operand with the file selected by HiRDB. If they match, HiRDB copies the contents of the normal status file to secondary File A and File B, then HiRDB switches the spare file in as the current file and starts the unit. If the files do not match, HiRDB is not started.

Conditions

The following operands must be specified:

- `pd_sts_initial_error = continue` or `excontinue`
- `pd_sts_last_active_file`

5.2.20 Operands related to work table files

66) pdwork -v "*HiRDB-file-system-area-name*"[,"*HiRDB-file-system-area-name*"]...

~<path name of up to 141 characters>

Specifies the names of HiRDB file system areas for work table files. Work table files are used for temporary storage of information during execution of SQL statements; they are created automatically by HiRDB. For details about the SQL statements that require work table files, see *Overview of the work table file* in the *HiRDB Version 9 Installation and Design Guide*.

You must not omit this operand, because doing so might prevent execution of SQL code that requires work table files.

Notes

- Specify in this operand the HiRDB file system area that was initialized using the `pdfmkfs` command.
- If the size of the work table file is large, specify a large HiRDB file system area. For details about how to estimate the work table file size, see the *HiRDB Version 9 Installation and Design Guide*.

  When an HiRDB file system area is set up initially with the `pdfmkfs -a` command, HiRDB will extend the area automatically whenever the amount of space specified in the `-n` option is used up. For details about the `pdfmkfs` command, see the manual *HiRDB Version 9 Command Reference*.
- The HiRDB file system areas for work table files must be different from the HiRDB file system areas for system files and RDAREAs.
- If more than one HiRDB file system area is specified in this operand, and one of these HiRDB file system areas generates an error when an attempt is made to create a work table file in it, that file system area will usually not be used thereafter. Instead, only the other specified HiRDB file system areas will be used.

  However, if subsequent attempts to create work table files fail in all of the other specified HiRDB file system areas (due to reasons such as insufficient space or an excessive numbers of files), creation of a work table file will be attempted in the HiRDB file system area that had been taken out of use. If a work table file can be created in it normally, that HiRDB file system area will then be used.

  Note that when HiRDB is shut down and restarted, any HiRDB file system area that was not being used because it had generated an error during work table file creation becomes usable again.

Operand rules

- At least one HiRDB file system area must be specified.
- A maximum of 16 HiRDB file system areas can be specified.
- This operand can be specified only once in the single server definition. If it is specified more than once, the first specification is effective.

5.2.21 Operands related to system log file configuration

67) pdlogadfg -d sys -g *file-group-name* [ONL]

Specifies a file group for a system log file. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign system log files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies a name for the file group. All file group names must be unique within a server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 200 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 200 times.

Notes

Before adding, modifying, or deleting this operand when HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while HiRDB Datareplicator is running, its extraction process might fail.

68) pdlogadpf -d sys -g *file-group-name* -a "*system-log-file-name*" [-b "*system-log-file-name*"]

Specifies the system log files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file that comprises the file group. Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

-b "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file B when dual system log files are to be used (`pd_log_dual = Y` specified). If `pd_log_dual = Y` is not specified, the system log file name is invalid, even if it is specified.

Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\log01`, `C:\hirdb\sysfile` is not case sensitive, but `log01` is case sensitive.

Note

Before adding, modifying, or deleting this operand when HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while HiRDB Datareplicator is running, its extraction process might fail.

5.2.22 Operands related to synchronization point dump file configuration

69) pdlogadfg -d spd -g *file-group-name* [ONL]

Specifies a file group for synchronization point dump files. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign synchronization point dump files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<<identifier>((1-8 characters))

Specifies a name for the file group. All file group names must be unique within a server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 30 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 60 times.

70) pdlogadpf -d spd -g *file-group-name* -a "*synchronization-point-dump-file-name*" [-b "*synchronization-point-dump-file-name*"]

Specifies the synchronization point dump files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name*: ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*synchronization-point-dump-file-name*": ~<path name> ((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file that comprises the file group. Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

-b "*synchronization-point-dump-file-name*": ~<path name> ((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file B when dual synchronization point dump files are to be used (`pd_spd_dual = Y` specified). If `pd_spd_dual = Y` is not specified, the synchronization point dump file name is invalid, even if it is specified.

Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sync01`, `C:\hirdb\sysfile` is not case sensitive, but `sync01` is case sensitive.

5.2.23 Operands related to Plug-ins

71) pdplgprm -n *plug-in-name* [-s *shared-memory-size*]

Specifies the name of a plug-in and the size of the memory to be shared by the plug-in. Omit this operand if no plug-ins are to be used.

Conditions

The plug-in specified here must have been registered in HiRDB with the `pdplgrgst` command.

`-n` *plug-in-name*: ~<identifier>((1-30 characters))

For details about the names of plug-ins that can be specified here, see the manuals for the plug-ins.

-s *shared-memory-size*: ~<unsigned integer>((1-2000000))<<0>> (kilobytes)

Specifies in kilobytes the size of the shared memory to be used by the plug-in. For details about the size of the shared memory to be used by the plug-in, see the manual for the applicable plug-in.

Effects on individual estimation formulas

If the value of the `pdplgprm` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for shared memory used by a single server*

6 Front-End Server Definition

This chapter explains the operands of the front-end server definition.

6.1 Operand formats

A front-end server definition defines information for a front-end server. This section explains the formats used to specify the operands of a front-end server definition. Note that the numbers in the following table correspond to the numbers assigned to the explanations of the individual operands in *6.2 Operand explanations*.

For users who are creating HiRDB system definitions for the first time

The first step is to determine the values to be specified for the operands shown in boldface type. In principle, HiRDB can be started once the boldface operands have been specified.

No.	Format	Operand category
1	`[set pd_process_count =` *resident-processes-count*`[,`*resident-processes-count-at-server-startup* `]]`#	Processes
2	`[set pd_server_cleanup_interval =` *interval-for-stopping-nonresident-server-processes*`]`#	
3	`[set pd_optimize_level =` *SQL-optimization-option* `[,`*SQL-optimization-option*`]...]`#	SQL optimization
4	`[set pd_additional_optimize_level =` *SQL-extension-optimizing-option* `[,`*SQL-extension-optimizing-option*`]...]`#	
5	`[set pd_floatable_bes = "`*back-end-server-name*`"[,"`*back-end-server-name*`"]...]`	
6	`[set pd_non_floatable_bes = "`*back-end-server-name*`"[,"`*back-end-server-name*`"]...]`	
7	`[set pd_watch_pc_client_time =` *maximum-client-request-wait-time*`]`*	System monitoring
8	`[set pd_spd_syncpoint_skip_limit =` *maximum-number-of-skipped-synchronization-point-dumps*`]`#	
9	`[set pd_cwaittime_wrn_pnt =` *output-condition-for-SQL-runtime-warning-information (% specification)* `|` *output-condition-for-SQL-runtime-warning-information (time specification)*`]`#	SQL runtime warning output facility
10	`[set pd_uap_exerror_log_use = YES | NO]`#	Facility for output of extended SQL error information
11	`[set pd_fes_lck_pool_size =` *front-end-server-lock-pool-size*`]`#	Lock
12	`[set pd_fes_lck_pool_partition =` *front-end-server-lock-pool-partition-count*`]`#	
13	`[set pd_lck_hash_entry =` *lock-pool-hash-entry-count*`]`#	
14	`[set pd_sql_object_cache_size =` *SQL-object-buffer-size*`]`#	Buffers
15	`[set pd_table_def_cache_size =` *table-definition-information-buffer-size*`]`#	
16	`[set pd_auth_cache_size =` *user-privilege-information-buffer-size*`]`#	
17	`[set pd_view_def_cache_size =` *view-analysis-information-buffer-size*`]`#	
18	`[set pd_type_def_cache_size =` *user-defined-type-information-buffer-size*`]`#	
19	`[set pd_routine_def_cache_size =` *routine-definition-information-buffer-size*`]`#	
20	`[set pd_registry_cache_size =` *registry-information-buffer-size*`]`#	
21	`[set pd_rpc_trace = Y | N]`#	RPC trace information
22	`[set pd_rpc_trace_name = "`*name-for-RPC-trace-collection-files*`"]`#	

No.	Format	Operand category
23	`[set pd_rpc_trace_size =` *RPC-trace-collection-file-size*`]`#	
24	`[set pd_module_trace_max =` *maximum-number-of-module-traces-that-can-be-stored*`]`#	Troubleshooting information
25	`[set pd_module_trace_timer_level = 0 | 10 | 20]`#	
26	`[set pd_pth_trace_max =` *maximum-number-of-stored-communication-traces*`]`#	
27	`[set pd_audit_def_buffer_size =` *security-audit-information-buffer-length*`]`	Security
28	`[set pd_java_stdout_file = "`*Java-virtual-machine-standard-output-and-standard-error-output-destination-file*`"]`#	Java
29	`[set pd_java_castoff = Y | N]`#	
30	`[set pd_log_dual = Y | N]`#	System log files
31	`[set pd_log_remain_space_check = warn | safe]`#	
32	`[set pd_log_auto_unload_path = "`*unload-log-file-output-directory*`" [,"`*unload-log-file-output-directory*`"]...]`	
33	`[set pd_log_auto_unload_restart = Y | N]`#	
34	`[set pd_log_singleoperation = Y | N]`#	
35	`[set pd_log_rerun_reserved_file_open = Y | N]`#	
36	`[set pd_log_rerun_swap = Y | N]`#	
37	`[set pd_log_swap_timeout =` *wait-time-for-completion-of-system-log-file-swapping*`]`#	
38	`[set pd_log_unload_check = Y | N]`#	
39	`[set pd_log_max_data_size =` *log-input/output-buffer-size*`]`#	
40	`[set pd_log_write_buff_count =` *log-output-buffer-sectors-count*`]`#	
41	`[set pd_log_rec_leng =` *system-log-file-record-length*`]`#	
42	`[set pd_log_rollback_buff_count =` *rollback-log-input-buffer-sector-count*`]`#	
43	`[set pd_log_auto_expand_size =` *extension-amount-per-system-log-file-extension-trigger*`[,`*extension-limit*`]]`#	
44	`[set pd_spd_dual = Y | N]`#	Synchronization point dump files
45	`[set pd_spd_assurance_msg = Y | N]`#	
46	`[set pd_spd_assurance_count =` *number-of-guaranteed-valid-generations*`]`#	
47	`[set pd_spd_reduced_mode =` *reduced-mode-operation-option*`]`#	
48	`[set pd_spd_reserved_file_auto_open = Y | N]`#	
49	`[set pd_spd_max_data_size =` *synchronization-point-dump-file-buffer-size*`]`#	
50	`[set pd_log_sdinterval =` *system-log-output-volume*`[,`*interval*`]]`#	
51	**set pd_sts_file_name_1 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"**	Server status files
	[set pd_sts_file_name_2 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	

No.	Format	Operand category

	`[set pd_sts_file_name_3 = "`*logical-file-name*`"`
	`,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	

	`[set pd_sts_file_name_4 = "`*logical-file-name*`"`
	`,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	

	`[set pd_sts_file_name_5 = "`*logical-file-name*`"`
	`,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	

	`[set pd_sts_file_name_6 = "`*logical-file-name*`"`
	`,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	

	`[set pd_sts_file_name_7 = "`*logical-file-name*`"`
	`,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	

52	`[set pd_sts_initial_error = stop	continue	excontinue]`#	Server status files (when an error occurs)
53	`[set pd_sts_singleoperation = stop	continue]`#	
54	`[set pd_sts_last_active_file = "`*logical-file-name*`"]`	
55	`[set pd_sts_last_active_side = A	B]`	
56	**{{pdlogadfg -d sys -g *file-group-name* [ONL]}}**	System log file configuration
57	**{{pdlogadpf -d sys -g *file-group-name*** **-a "*system-log-file-name*" [-b "*system-log-file-name*"]}}**	
58	**{{pdlogadfg -d spd -g *file-group-name* [ONL]}}**	Synchronization point dump file configuration
59	**{{pdlogadpf -d spd -g *file-group-name*** **-a "*synchronization-point-dump-file-name*"** **[-b "*synchronization-point-dump-file-name*"]}}**	
60	`{{[pdplgprm -n` *plug-in-name* `[-s` *shared-memory-size*`]]}}`	Plug-ins

#: If this operand is omitted, the value specified in the same operand in the server common definition is used. However, for the following operands, the value specified for the same operand in the system common definition, rather than the server common definition, is used:

- `pd_optimize_level`
- `pd_additional_optimize_level`
- `pd_cwaittime_wrn_pnt`
- `pd_uap_exerror_log_use`

6.2 Operand explanations

6.2.1 Operands related to processes

1) pd_process_count = ***resident-processes-count*****[,*****resident-processes-count-at-server-startup*****]**

~<unsigned integer>((0-2000))

resident-processes-count

Specifies the number of processes that can be made resident in the front-end server. A resident process is a process that is activated at the time the server is started.

Advantage

By activating the processes used by transactions that can be processed concurrently by the front-end server at the time of system startup and keeping them resident, the process startup time can be reduced even when new transactions are entered. However, HiRDB startup will take longer.

Specification guidelines

- Determine the value to be specified based on the process private area allocated to the front-end server's server process and the amount of real memory in the processor. For details about the process private area of the server process, see the *HiRDB Version 9 Installation and Design Guide*.
- In a multiple front-end server configuration, if the `pd_max_bes_process` or `pd_max_dic_process` operand has also been specified, the value specified for this operand must satisfy the following condition:

  Value of pd_process_count ≤ *value of pd_max_bes_process or pd_max_dic_process*
- For this operand, specify a value that is no more than the front-end server's maximum number of processes that can be activated.

Server type	Maximum number of processes that can be activated
Multiple front-end servers	*Value of pd_max_users* + 1
Other than multiple front-end servers	*Value of pd_max_users*

Tuning the specified value

For details about how to tune the resident processes count, see the *HiRDB Version 9 System Operation Guide*.

Notes

- Because the number of resident processes has a direct effect on the availability of memory space and on the CPU, specifying an unnecessarily large number might prevent HiRDB from starting or might degrade the server machine's processing performance.
- If more processes than the resident process count are needed, additional processes are dynamically started, up to the maximum processes count allowed. However, depending on the value specified for the `pd_max_server_process` operand, it might not be possible to start all of the processes indicated by the maximum processes count.

Operand default value

If this operand is omitted (or `0` is specified), the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the following value is assumed:

↑ *maximum number of processes that can be activated* ÷ 2 ↑

resident-processes-count-at-server-startup

Specifies the number of processes that can be made resident during HiRDB startup.

It is common for resident processes to be activated during HiRDB startup. If there are many resident processes, the amount of time required for HiRDB startup increases proportionately. For example, it takes approximately 1 second to activate a process on a server machine with a 100-MIPS performance rating.

The differences in processing that result depending on whether a resident processes count at server startup is specified are as follows:

- When there is no specification of a resident processes count at server startup (when, for example, `pd_process_count = 500` is specified)

  All 500 resident processes are activated during HiRDB startup, and HiRDB will not start until they are all activated. In this example, it would take a 100-MIPS server machine about 500 seconds to activate all the resident processes during HiRDB startup.

- When a resident processes count at server startup is specified (when, for example, `pd_process_count = 500, 50` is specified)

  Some of the resident processes (50 in this case) are activated during HiRDB startup, and the remaining resident processes are activated after HiRDB startup. HiRDB can be started as long as the specified number of resident processes are activated. In this example, it would take a 100-MIPS server machine about 50 seconds to activate 50 resident processes during HiRDB startup. The remaining 450 resident processes (in this example) would be activated after HiRDB startup.

Advantage

The amount of time required for HiRDB startup is reduced. Use this option when you want to reduce the HiRDB startup time as much as possible, such as when you are using the system switchover facility.

Specification guideline

Specify a value equal to the number of processes that will be required immediately after HiRDB startup is completed.

Notes

When you specify a resident processes count at server startup, recheck the value in the `PDCWAITTIME` operand of the client environment definition.

If more UAPs than the resident processes count at server startup are to be executed immediately following HiRDB startup, transaction processing might not be performed until after the remaining resident processes have been activated. Therefore, if the value specified in the `PDCWAITTIME` operand of the client environment definition is small, it might not be possible to process some UAPs due to timeouts. For details about the `PDCWAITTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.

2) pd_server_cleanup_interval = *interval-for-stopping-nonresident-server-processes*

~<unsigned integer>((0-1440)) (minutes)

Specifies in minutes the interval for checking for nonresident server processes in HiRDB that are to be stopped. The facility for stopping nonresident server processes is applied when the number of executing server processes exceeds the number of processes that can be made resident (value specified by the `pd_process_count` operand). The number of server processes that the facility stops is computed automatically by HiRDB.

Advantages

This facility improves the utilization rates of the memory and of process resources because it increases the number of nonresident processes that can be reused when the workload (number of server processes that are executing) peaks.

Specification guidelines

- For example, if there is only one hour a day during which peak workload occurs and the intervals between peaks within that hour are approximately two minutes apart, specify `2` for this operand.
- This facility does not have any effect if the number of server processes that execute concurrently during peak periods is always fewer than the number of resident processes. In this case, omit this operand.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `0` is assumed.

Tuning the specified value

Collect statistical information on system operations for each server for one week. Determine the workload peaks from the number of server processes being serviced (`# OF PROCESSES ON SERVICE`). If that peak value exceeds the currently set number of processes that can be made resident (value specified by the `pd_process_count` operand), determine the interval between individual peaks and set that number of minutes.

However, if there are ample resources, such as memory and CPU, in the server machine, adding the shortfall in the number of processes to the number of resident processes (in other words, increasing the value of the `pd_process_count` operand) is more effective in improving performance than specifying the `pd_server_cleanup_interval` operand.

Note

If this operand is omitted or `0` is specified, the system checks every 10 seconds for nonresident server processes that are waiting to be serviced and stops any such processes that are found.

6.2.2 Operands related to SQL optimization

3) pd_optimize_level = ***SQL-optimization-option*** **[,*SQL-optimization-option*]...**

~<identifier or unsigned integer>

Specifies SQL optimization options. For details about the SQL optimization options, see the *HiRDB Version 9 UAP Development Guide.* The SQL optimization option facilities are explained as follows.

SQL optimization option facility	Identifier	Unsigned integer	S	P
Forced nest-loop-join	`"FORCE_NEST_JOIN"`	`4`	Y	Y
Making multiple SQL objects	`"SELECT_APSL"`	`10`	Y	Y
Increasing the target floatable servers (back-end servers for fetching data)	`"FLTS_INC_DATA_BES"`	`16`	N	Y
Prioritized nest-loop-join	`"PRIOR_NEST_JOIN"`	`32`	Y	Y
Increasing the number of floatable server candidates	`"FLTS_MAX_NUMBER"`	`64`	N	Y
Priority of OR multiple index use	`"PRIOR_OR_INDEXES"`	`128`	Y	Y
Group processing, ORDER BY processing, and DISTINCT set function processing at the local back-end server	`"SORT_DATA_BES"`	`256`	N	Y
Suppressing use of AND multiple indexes	`"DETER_AND_INDEXES"`	`512`	Y	Y
Rapid grouping facility	`"RAPID_GROUPING"`	`1024`	Y	Y
Limiting the target floatable servers (back-end servers for fetching data)	`"FLTS_ONLY_DATA_BES"`	`2048`	N	Y
Separating data collecting servers	`"FLTS_SEPARATE_COLLECT_SVR"`	`2064`	N	Y
Suppressing index use (forced table scan)	`"FORCE_TABLE_SCAN"`	`4096`	Y	Y
Forcing use of multiple indexes	`"FORCE_PLURAL_INDEXES"`	`32768`	Y	Y
Suppressing creation of update-SQL work tables	`"DETER_WORK_TABLE_FOR_UPDATE"`	`131072`	Y	Y
Deriving search acceleration condition	`"DERIVATIVE_COND"`	`262144`	Y	Y
Applying key condition that includes scalar operation	`"APPLY_ENHANCED_KEY_COND"`	`524288`	Y	Y
Facility for batch acquisition from functions provided by plug-ins	`"PICKUP_MULTIPLE_ROWS_PLUGIN"`	`1048576`	Y	Y
Facility for moving search conditions into a derived table	`"MOVE_UP_DERIVED_COND"`	`2097152`	Y	Y

S: HiRDB/Single Server

P: HiRDB/Parallel Server

Y: Specification is valid.

N: Specification is not applicable.

Operand specification methods

Select the SQL optimization options to be applied and specify their identifiers or unsigned integers. Although either identifiers or unsigned integers (computed values) can be used to specify options, the use of identifiers is usually recommended.

- Using identifiers

  To apply *forced nest-loop-join* and *making multiple SQL objects*, specify the following:

  `pd_optimize_level="FORCE_NEST_JOIN","SELECT_APSL"`
- Using unsigned integers

  To apply *forced nest-loop-join* and *making multiple SQL objects*, specify the following:

  `pd_optimize_level=4,10`
- When you have upgraded from HiRDB Version 5.0 or earlier version

  The total value specified in HiRDB Version 5.0 or earlier version is also valid. If there is no need to change the optimization option, you need not change the specification of this operand after upgrading to HiRDB Version 6 or a later version. To add the optimization option, use the following example:

  Example:

  Forced nest-loop-join and *making multiple SQL objects* have been applied to HiRDB Version 5.0, and *priority of OR multiple index use* is to be newly added.

  `pd_optimize_level = 14,128`

  However, because this specification makes it difficult to identify the facilities being applied, use of identifier specification is recommended.

Operand rules

- Identifiers and unsigned integers cannot be specified together.
- Identifier specification

  ● Enclose each SQL optimization option in quotation marks (").

  ● If no SQL optimization option is used, specify `NONE`. When `NONE` and an identifier are both specified, the specification of `NONE` is invalid.

  ● Identifiers can be specified in uppercase or lowercase letters.

  ● Specifying the same identifier more than once is the same as specifying it once.
- Unsigned integer specification

  ● If you do not use the SQL optimization option explained here, specify `0`. However, if both `0` and a non-zero unsigned integer are specified, the specification of `0` is ignored.

  ● Specifying the same unsigned integer more than once is the same as specifying it once.

Specification guidelines

For specification guidelines, see `PDSQLOPTLVL` in the *HiRDB Version 9 UAP Development Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the system common definition is assumed. If the same operand is also omitted in the system common definition, the following values are assumed:

- HiRDB/Single Server

```
"PRIOR_NEST_JOIN","PRIOR_OR_INDEXES","DETER_AND_INDEXES","RAPID_GROUPIN
G",
"DETER_WORK_TABLE_FOR_UPDATE","APPLY_ENHANCED_KEY_COND"
```

- HiRDB/Parallel Server

```
"PRIOR_NEST_JOIN","PRIOR_OR_INDEXES","SORT_DATA_BES","DETER_AND_INDEXES
",
"RAPID_GROUPING","DETER_WORK_TABLE_FOR_UPDATE","APPLY_ENHANCED_KEY_COND
"
```

However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `SELECT_APSL`.

Notes

- If SQL optimization is specified in an SQL statement, SQL optimization specification takes precedence over the specification in this operand. For details about SQL optimization specification, see the manual *HiRDB Version 9 SQL Reference*.
- If SQL optimization is specified in an SQL statement (`CREATE PROCEDURE`, `CREATE TYPE`, `ALTER PROCEDURE`, `CREATE TRIGGER`, `ALTER ROUTINE`, or `ALTER TRIGGER`) inside a stored routine or

trigger, the SQL optimization option inside the SQL statement takes precedence over the specification in this operand.

Relationship to other operands

- When the `pd_floatable_bes` or `pd_non_floatable_bes` operand is specified in the front-end server definition, specification of increasing the target floatable servers (back-end servers for fetching data) and limiting the target floatable servers (back-end servers for fetching data) is invalid.
- When `KEY` is specified for the `pd_indexlock_mode` operand, specification of suppressing creation of update-SQL work tables is invalid.

Relationship to client environment definition

The value of this operand can be different for each client. To change the value for a client, specify the `PDSQLOPTLVL` operand in the client environment definition. For details about the `PDSQLOPTLVL` operand, see the *HiRDB Version 9 UAP Development Guide*.

4) pd_additional_optimize_level = *SQL-extension-optimizing-option* [,*SQL-extension-optimizing-option*]...
~<identifier or unsigned integer>

Specifies SQL extension optimizing options. For details about the SQL extension optimizing options, see the *HiRDB Version 9 UAP Development Guide.*

The following table describes the SQL extension optimizing option facilities.

SQL extension optimizing option facility	Identifier	Unsigned integer
Application of optimizing mode 2 based on cost	`"COST_BASE_2"`	1
Hash execution of a hash join or a subquery#	`"APPLY_HASH_JOIN"`	2
Facility for applying join conditions that include value expressions#	`"APPLY_JOIN_COND_FOR_VALUE_EXP"`	32
Enabling substructure indexes for `XMLEXISTS` predicates that include parameters#	`"ENABLE_INDEX_XMLEXISTS_PARAM"`	256

#: These items are valid when *application of optimizing mode 2 based on cost* is specified.

Operand specification methods

Select the SQL extension optimization options to be applied and specify their identifiers or unsigned integers. Although either identifiers or unsigned integers (computed values) can be used to specify options, the use of identifiers is usually recommended.

- Using identifiers

  To apply *application of optimizing mode 2 based on cost* and *hash execution of a hash join or a subquery*, specify the following:

  ```
  pd_additional_optimize_level= "COST_BASE_2", "APPLY_HASH_JOIN"
  ```

- Using unsigned integers

  To apply *application of optimizing mode 2 based on cost* and *hash execution of a hash join or a subquery*, specify the following:

  ```
  pd_additional_optimize_level=1,2
  ```

Operand rules

- Identifiers and unsigned integers cannot be specified together.
- Identifier specification
  - Enclose each SQL extension optimizing option in quotation marks (").
  - If you do not use the SQL extension optimizing option explained here, specify `NONE`. However, if both `NONE` and an identifier other than `NONE` are specified, the specification of `NONE` is invalid.
  - Identifiers can be specified in uppercase or lowercase letters.
  - Specifying the same identifier more than once is the same as specifying it once.
- Unsigned integer specification
  - If you do not use the SQL extension optimizing option explained here, specify `0`. However, if both `0` and a non-zero unsigned integer are specified, the specification of `0` is invalid.
  - Specifying the same unsigned integer more than once is the same as specifying it once.

Specification guidelines

For specification guidelines, see PDADDITIONALOPTLVL in the *HiRDB Version 9 UAP Development Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the system common definition is assumed. If the same operand is also omitted in the system common definition, the default is COST_BASE_2, unless v6compatible has been specified in the pd_sysdef_default_option operand, in which case the default is NONE.

Notes

- If the SQL optimization is specified in an SQL statement, SQL optimization specification takes precedence over the specification in this operand. For details about SQL optimization specification, see the *HiRDB Version 9 SQL Reference*.
- If the SQL extension optimizing option is specified in an SQL statement (CREATE PROCEDURE, CREATE TYPE, CREATE TRIGGER, ALTER PROCEDURE, ALTER ROUTINE, or ALTER TRIGGER) inside a stored routine or trigger, the SQL extension optimizing option inside the SQL statement takes precedence over the specification in this operand.

Relationship to client environment definition

The value of this operand can be different for each client. To change the value for a client, specify the PDADDITIONALOPTLVL operand in the client environment definition. For details about the PDADDITIONALOPTLVL operand, see the *HiRDB Version 9 UAP Development Guide.*

5) pd_floatable_bes = "*back-end-server-name*"[,"*back-end-server-name*"]...

~<identifier>((1-8 characters))

Specifies back-end servers to be used as floating servers.

Not all of the specified back-end servers can actually be used as floating servers.

Normally, servers used for data extraction are not used as floating servers. However, this operand enables servers that are used for data extraction to also become floating server candidates.

Operand rules

- If an undefined back-end server name is specified, it is ignored.
- If all the specified back-end server names are undefined, this entire operand is ignored.

Relationship to other operands

- If the pd_non_floatable_bes operand is also specified, it is invalidated.
- If the pd_floatable_bes operand is specified, *increasing the target floatable servers (back-end servers for fetching data)* of the pd_optimize_level operand is not applied.
- If the pd_floatable_bes operand is specified, *limiting the target floatable servers (back-end servers for fetching data)* of the pd_optimize_level operand is not applied.

6) pd_non_floatable_bes = "*back-end-server-name*"[,"*back-end-server-name*"]...

~<identifier>((1-8 characters))

Specifies back-end servers that are not to be used as floating servers.

If all back-end servers are specified, this operand is invalid.

Operand rule

If an undefined back-end server name is specified, it is ignored.

Relationship to other operands

- If the pd_floatable_bes operand is also specified, it is invalidated.
- If the pd_non_floatable_bes operand is specified, *increasing the target floatable servers (back-end servers for fetching data)* of the pd_optimize_level operand is not applied.
- If the pd_non_floatable_bes operand is specified, *limiting the target floatable servers (back-end servers for fetching data)* of the pd_optimize_level operand is not applied.

6.2.3 Operands related to system monitoring

7) pd_watch_pc_client_time = *maximum-client-request-wait-time*

~<unsigned integer>((0-65535)) (seconds)

Specifies in seconds the maximum amount of time for a server to wait for the next request from a HiRDB client after the HiRDB server returns a response to a request from a Windows-compatible HiRDB client.

If no request comes from the HiRDB client within the specified amount of time, it will be assumed that an error occurred at the client and the connection between the server and the client will be terminated forcibly. No notice of disconnection is sent to the HiRDB client in such a case.

The time that is monitored is the period between `CONNECT` and `DISCONNECT` (that is, the non-transaction status time), excluding the period between SQL execution startup and `COMMIT` or `ROLLBACK`.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `3600` is assumed.

Notes

- When `0` is specified, the server waits indefinitely for the next request from the HiRDB client.
- If a small value (up to `600`, for example) is specified for this operand, the HiRDB client might detect *server down* during SQL execution and might not terminate correctly.
- For a UNIX edition of a HiRDB client (including a Linux edition of a HiRDB client), time is not monitored regardless of the value specified for this operand. To monitor time for a UNIX edition of a HiRDB client, specify the `PDSWATCHTIME` client environment definition of the HiRDB client.

Relationship to client environment definition

The value of this operand can be modified for each client. To do so, the `PDSWATCHTIME` operand must be specified in the client environment definition. For details about the `PDSWATCHTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.

8) pd_spd_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps*

~<unsigned integer>((0, 2-100000))

If an event such as an endless loop occurs in a UAP, acquisition of synchronization point dumps might be skipped. If several synchronization point dumps are not acquired (instead, they are skipped), there will be an increase in the number of overwrite-disabled system log files, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

If HiRDB is forcibly terminated or terminates abnormally when the number of system log files that cannot be overwritten has reached one-half or more of all system log files, a shortage of system log files occurs during rollback processing when HiRDB is restarted. In this case, HiRDB cannot be restarted unless new system log files are added. Any such restart processing will take a longer time than normal.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction). When the number of skipped synchronization point dumps reaches the specified operand value, the target transaction is cancelled forcibly and rolled back. This is called the *skipped effective synchronization point dump monitoring facility*.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

Advantage

Specifying this operand provides a means of dealing with endless loops in UAPs.

Specification guidelines

- Normally, specify `0` for this operand. When `0` is specified, HiRDB computes the upper limit for the skip count. If specifying `0` causes a problem or the `KFPS02101-I` message is issued, change the value of this operand. For guidelines on the value to specify, see the *HiRDB Version 9 System Operation Guide*.
- If the specified value is too large, all system log files might be placed in overwrite disabled status. If this happens, HiRDB terminates abnormally.
- If the specified value is too small, the number of transactions that are forcibly rolled back might increase.
- Specify this value taking into account the value of `pd_log_sdinterval` and the number of times synchronization point dumps are acquired when the transaction with the longest execution time and largest log output volume is processed.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the skipped effective synchronization point dump monitoring facility is not used.

Notes

- The monitoring provided for by the specification of this operand begins the first time a synchronization point dump is collected after HiRDB is started (including a restart).
- When this operand is specified, UAPs that are being executed in the no-log mode are also monitored. If the processing of a UAP being executed in the no-log mode is cancelled, the database cannot be automatically recovered, and thus RDAREAs are placed in the error shutdown state. Therefore, when specifying the upper limit, account also for the size of the system logs that are output from other transactions inside the server during the processing of UAPs that are executed in the no-log mode.
- If a transaction is delayed due to a high workload, the skip count increases because a synchronization point dump cannot be acquired until the delayed transaction is completed.
- If the skip count exceeds the upper limit, any process executing a transaction is forcibly terminated. In this case, the rollback logs, which are the logs output by these transactions, are output after the forced termination.
- The `pdload`, `pdmod`, `pdrorg`, `pdexp`, `pddbst`, `pdgetcst`, `pdrbal`, `pdvrup`, `pdmemdb`, and `pdextfunc` commands are not monitored by this facility.

6.2.4 Operands related to SQL runtime warning output facility

9) pd_cwaittime_wrn_pnt = *output-condition-for-SQL-runtime-warning-information (% specification)* | *output-condition-for-SQL-runtime-warning-information (time specification)*

Specify this operand when using the SQL runtime warning output facility. You can specify this operand using one of the following two methods:

- Specifying a percentage
- Specifying a time duration

For details about the SQL runtime warning output facility, see the *HiRDB Version 9 System Operation Guide*.

***output-condition-for-SQL-runtime-warning-information (% specification)*: ~<unsigned integer>((0-99)) or <unsigned decimal number>((0-99.999999)) (%)**

Specifies the condition for outputting SQL runtime warning information of the SQL runtime warning output facility as a percentage of the maximum client wait time (value of the `PDCWAITTIME` operand in the client environment definition). After SQL execution, HiRDB checks the SQL execution time. If this SQL execution time is determined to exceed the time specified by this operand, the following warning information is output. This is called the *SQL runtime warning output facility*.

- SQL runtime warning information file
- Warning message (`KFPA20009-W`)

Specification guidelines

Specify this operand as a percentage (%) of the value of the `PDCWAITTIME` operand. For example, if `100` (seconds) is specified for the `PDCWAITTIME` operand and `90` (%) is specified for the `pd_cwaittime_wrn_pnt` operand, HiRDB checks the SQL execution time after SQL execution. If the determined SQL execution time is 90 seconds or longer but less than 100 seconds, the warning information is output.

Example

```
PDCWAITTIME = 100
pd_cwaittime_wrn_pnt = 90
```

***output-condition-for-SQL-runtime-warning-information (time specification)*: ~<unsigned decimal number>((0-65534.999999))sec (seconds)**

Specifies the condition for outputting SQL runtime warning information of the SQL runtime warning output facility as a time duration. After SQL execution, HiRDB checks the SQL execution time. If this SQL execution time is determined to exceed the time specified by this operand, the following warning information is output. This is called the *SQL runtime warning output facility*.

- SQL runtime warning information file
- Warning message (`KFPA20009-W`)

Operand specification method

Specify the time duration (in seconds) to be used as the output trigger. (You can specify up to the sixth decimal.) Add `sec` to the specified value.

Example

```
pd_cwaittime_wrn_pnt = 0.001sec
```

The following explanation is the same for both percentage and time duration specification.

Operand rule

If `0` or `0sec` is specified in this operand, no warning information is output. (The SQL runtime warning output facility is not used.)

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, the following value is assumed.

- *output-condition-for-SQL-runtime-warning-information (% specification)*: `0`
- *output-condition-for-SQL-runtime-warning-information (time specification)*: `0sec`

Relationship to client environment definition

You can change the value of this operand for each client. To change it for each client, specify the `PDCWAITTIMEWRNPNT` operand of the client environment definition. For details about the `PDCWAITTIMEWRNPNT` operand, see the *HiRDB Version 9 UAP Development Guide*.

Relationship to other operands

This operand is related to the following operands:

- `pd_cwaittime_report_dir`
- `pd_cwaittime_report_size`

6.2.5 Operand related to the facility for output of extended SQL error information

10) pd_uap_exerror_log_use = YES | NO

Specifies whether to use the facility for output of extended SQL error information. For details about this facility, see the *HiRDB Version 9 UAP Development Guide*.

`YES`:

The facility for output of extended SQL error information is used. SQL error information is output to a client error log file and the SQL error report file.

`NO`:

The facility for output of extended SQL error information is not used.

Specification guidelines

If you manage SQL error information centrally, or output the SQL statements and the parameter information resulting from an error, we recommend that you specify `YES`.

Operand default value

If this operand is omitted, the value specified for the same operand in the system common definition takes effect. If the same operand is also omitted from the system common definition, `NO` is assumed.

Relationship to client environment definition

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDUAPEXERLOGUSE` operand in the client environment definition. If both this operand and the `PDUAPEXERLOGUSE` operand in the client environment definition are specified, the `PDUAPEXERLOGUSE` operand takes precedence.

For details about the `PDUAPEXERLOGUSE` operand, see the *HiRDB Version 9 UAP Development Guide*.

Effects on individual estimation formulas

If the value of the `pd_uap_exerror_log_use` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the size of the memory required to use the facility for output of extended SQL error information* under *Estimating the memory size required for a HiRDB/Parallel Server*

6.2.6 Operands related to lock

11) pd_fes_lck_pool_size = ***front-end-server-lock-pool-size***

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-2000000))**
- 64-bit mode: **((1-2000000000))**

Specifies in kilobytes the size of the shared memory area to be used by the front-end server for locking (lock pool).

The area specified by this operand is used for tables for managing locked resources. Note the following relationship between the number of tables for managing locked resources, the number of lock requests, and the lock pool size:

Number of tables for managing locked resources = number of lock requests = size of lock pool × *coefficient*

Specification guidelines

- In the 32-bit mode, the amount of lock pool space required for six lock requests is 1 kilobyte.
- In the 64-bit mode, the amount of lock pool space required for four lock requests is 1 kilobyte.
- Use the following formula to determine the value of this operand:

  $\uparrow\uparrow(a + b) \div$ *value of pd_fes_lck_pool_partition* $\uparrow \div c \uparrow \times$ *value of pd_fes_lck_pool_partition* (kilobytes)

a: Total number of transaction lock requests to be executed concurrently by the front-end server. The number of lock requests depends on the SQL. For details about how to determine the total number of lock requests, see *D. Determining the Number of Locked Resources*.

b: (*value of* `pd_max_users` + 3) × (*value of* `pd_max_access_tables` + 4)

c: Use `6` and `4` for 32- and 64-bit modes, respectively.

Tuning the specified value

See the usage rate for the locked resources management table (`%OF USE LOCK TABLE`) displayed for the front-end server in the statistical information on system operation by the statistics analysis utility. If the maximum usage rate equals or exceeds 80%, it is recommended that the operand's value be increased in preparation for future database expansion. If the maximum usage rate does not exceed 10%, it is recommended that the operand's value be decreased to conserve shared memory space.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is the value shown below:

● 32-bit mode

{(*pd_max_users value* + 3) × (*pd_max_access_tables value* + 4)} ÷ 6

● 64-bit mode

{(*pd_max_users value* + 3) × (*pd_max_access_tables value* + 4)} ÷ 4

Note

- If the value specified for this operand is too small, an SQL statement might return an error.
- A large value will increase the size of the shared memory used by HiRDB, which might cause a shortage of shared memory and prevent HiRDB from starting.

Relationship to other operands

This operand is related to the `pd_fes_lck_pool_partition` operand.

12) pd_fes_lck_pool_partition = *front-end-server-lock-pool-partition-count*

~<unsigned integer>((1-5000))

Specifies the number of lock pool partitions to be used in locking by the front-end server when distributing lock processing.

For details about distributing lock processing, see the *HiRDB Version 9 System Operation Guide*.

Tuning the specified value

For details about how to tune the number of lock pool partitions, see the *HiRDB Version 9 System Operation Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `1`.

Notes

- When the value set in this operand is too large, the size of the shared memory assigned to each lock pool partition becomes smaller, which might result in not enough lock pool partition capacity, causing SQL to return an error. Lock processing also takes time, so system performance declines. If this becomes an issue, specify a smaller value in this operand.
- The lock pool size must be at least 1 kilobyte. If a value larger than the value of `pd_lck_pool_size` is specified, the `KFPS00421-W` message will be issued and the value of `pd_lck_pool_size` will be assumed for this operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_fes_lck_pool_size`
- `pd_lck_deadlock_check_interval`

Effects on individual estimation formulas

If the value of the `pd_fes_lck_pool_partition` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

13) pd_lck_hash_entry = *lock-pool-hash-entry-count*

~<unsigned integer>((0-2147483647))

Specifies the number of hash table entries to be used in the lock pool. According to the value specified here, HiRDB allocates a lock pool in the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand.

Consider specifying a value for this operand in the following cases:

- If you do not wish to change the shared memory size if possible when upgrading to version 06-02 or later, specify `11261`. In this case, the same number of hash entries is allocated as in the earlier version, and the hash table size inside the lock pool remains the same as before.
- It is possible to improve performance by specifying in this operand a value greater than the recommended value shown below. However, specifying a value greater than variable *a* (also shown below) will not improve performance over the case in which *a* is specified.

  The recommended value is as follows:

  Recommended value = Largest prime number not exceeding MAX($\uparrow a \div 10 \uparrow$, 11,261)

Variable	Formula for computing the variable	
a	If `pd_fes_lck_pool_size` is omitted	(*b* + 3) × (*pd_max_access_tables value* + 4)
	If `pd_fes_lck_pool_size` is specified	*pd_fes_lck_pool_size value* × c
b	Multiple front-end server	*pd_max_users value* + 1
	Not multiple front-end server	*pd_max_users value*

Variable	Formula for computing the variable
c	6 for the 32-bit mode; 4 for the 64-bit mode

Operand rules

- If this operand and the `pd_lck_hash_entry` operand of the server common definition are both omitted or `0` is specified in this operand, HiRDB calculates a recommended value for the server. (However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default for this operand is `11261`.)
- When a value that is neither `0` nor a prime number is specified in this operand, HiRDB assumes that the specification is the largest prime number that does not exceed the specified value.

Note

If the value specified in this operand is too small, there might be an insufficient number of hash entries, and performance might deteriorate. If this operand is omitted, there will never be a shortage of hash entries and performance will not deteriorate due to an insufficient number of hash entries.

6.2.7 Operands related to buffers

14) pd_sql_object_cache_size = *SQL-object-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((22-256000))**
- 64-bit mode: **((22-2000000))**

Specifies in kilobytes the size of the buffer area (shared memory) in which SQL objects are to be placed.

Specification guidelines

- SQL objects are saved in a buffer until the user's transaction has terminated. The buffer must be large enough to store the SQL objects of all transactions that will be executed concurrently.
- When the hit rate is low for SQL object buffers, performance might be degraded due to the overhead required for SQL parsing processing.
- SQL analysis can be reduced by saving the SQL objects of static SQLs in the buffer after transaction termination (until the buffer runs out of space) and sharing them among multiple users who execute the same UAP. To effectively utilize the buffer, allocate it so that the SQL objects of frequently-used UAPs are resident in the buffer.
- To estimate the buffer size, first determine the buffer size needed for UAP execution from the length of the SQL objects from the SQL statements to be issued by the UAP. Then, compute the buffer size by considering the number of UAPs that will be executed concurrently and the number of concurrently executing users.
- For details about how to estimate the length of the SQL object from a single SQL statement, see *C.2 Formulas for determining size of SQL object buffer (pd_sql_object_cache_size).*

Operand default

If this operand is omitted, the specification of the same operand in the server common definition or in the system common definition, in that order, is assumed. If the same operand is also omitted in the server common definition and the system common definition, the following value is assumed:

(*value of pd_max_users* + 3) × 22

Tuning the specified value

For details about tuning the SQL object buffer size, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_sql_object_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

15) pd_table_def_cache_size = *table-definition-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((100-65535))**
- 64-bit mode: **((100-2000000))**

Specifies in kilobytes the size of the buffer (shared memory) for table and sequence generator definition information. This information is used during SQL statement pre-processing. Definition information stored in this buffer is managed by the LRU method.

Advantages

- Table and sequence generator definition information that has been used is retained in memory as long as possible so that it can be used again without an input operation.
- Performance improves when a large number of dynamic SQLs are used.
- The number of communications with the dictionary server is reduced.

Specification guidelines

- Specify the total size of the definition information needed for frequently-used tables and sequence generators.
- The table definition information buffer size per sequence generator is 8 kilobytes.
- For details about determining the size of the table definition information buffer per table, see *C.4 Formulas for determining size of table definition information buffer (pd_table_def_cache_size)*. For temporary tables, determine the size in the same manner as for non-partitioned tables.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is the following value:

《↑ $\sqrt{\text{(}\textit{value of pd\_max\_bes\_process or pd\_max\_dic\_process, whichever is larger}\text{)}}$ ↑ × 100 + 3》

Tuning the specified value

For details about how to tune the size of the table definition information buffer, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_table_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

16) pd_auth_cache_size = *user-privilege-information-buffer-size*

~<unsigned integer>((1-100)) (kilobytes)

Specifies in kilobytes the size of the buffer (shared memory) for user privilege information.

Specification guidelines

- The user privilege information buffer stores CONNECT privilege, DBA privilege, and audit privilege information. If this buffer contains no information, information is obtained from a dictionary table during HiRDB connection, thus lengthening the response time. Therefore, specify a buffer size that can store the information for the users who are always connected.
- Storing the user privilege information of each user requires 68 bytes. Use this information when computing the total buffer size.

Tuning the specified value

For details about how to tune the size of the user privilege information buffer, see the *HiRDB Version 9 System Operation Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `1`.

Effects on individual estimation formulas

If the value of the `pd_auth_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

17) pd_view_def_cache_size = ***view-analysis-information-buffer-size***

~<unsigned integer> (kilobytes)

- 32-bit mode: **((0-32000))**
- 64-bit mode: **((0-2000000))**

Specifies in kilobytes the size of the buffer (shared memory) for view analysis information.

Advantage

View analysis information that has been used is kept in the shared memory and can be used subsequently without an I/O operation.

Specification guideline

The total size of the view analysis information for frequently-used view tables is specified. For details about determining the size of the view analysis information buffer, see *C.3 Formulas for determining size of view analysis information buffers (pd_view_def_cache_size)*.

Tuning the specified value

For details about how to tune the size of the buffer for view analysis information, see the *HiRDB Version 9 System Operation Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is the following value:

《$\uparrow \sqrt{(\textit{value of pd\_max\_users} + 3)} \uparrow \times 8$》

Effects on individual estimation formulas

If the value of the `pd_view_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

18) pd_type_def_cache_size = ***user-defined-type-information-buffer-size***

~<unsigned integer> (kilobytes)

- 32-bit mode: **((100-65536))**
- 64-bit mode: **((100-2000000))**

This operand is applicable to user-defined types; specification of this operand is recommended when user-defined types will be used.

Specifies in kilobytes the size of the buffer (shared memory) for information on user-defined types. When a user-defined type is used, information on it is stored in this buffer. This information is used during pre-processing of SQL statements.

Advantages

- When user-defined type information is stored in this buffer, it is not necessary to access the dictionary table when the same user-defined type is used subsequently, thus reducing the number of input operations and the CPU usage time.
- The number of communications with the dictionary server is reduced.
- Specification of this operand improves performance when many dynamic SQLs are used.

Specification guidelines

Specify the total size of the definition information for frequently-used user-defined types that are defined in tables. The following formula can be used to determine the definition information size for one user-defined type:

$\uparrow \{((0.3 + 0.2 \times a + 0.1 \times b) + 3) \div 4\} \uparrow \times 4$ (kilobytes)

a: Number of user-defined type attributes *b*: Number of subtypes that have inherited a supertype

Tuning the specified value

For details about how to tune the size of the buffer for user-defined type information, see the *HiRDB Version 9 System Operation Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `0`.

Effects on individual estimation formulas

If the value of the `pd_type_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

19) pd_routine_def_cache_size = *routine-definition-information-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((0, 20-65536))**
- 64-bit mode: **((0, 20-2000000))**

Specifies in kilobytes the size of the buffer (shared memory) for storing the following types of definition information (this information is used during pre-processing of SQL statements):

- Plug-in facility definition information
- System definition scalar facility definition information
- Routine definition information

Advantages

- When these types of definition information are stored in this buffer, it is not necessary to access the dictionary table when the same information is used subsequently, thus reducing the number of I/O operations and CPU usage time.
- The number of communications with the dictionary server is reduced.

Application criteria

Specify this operand when a large number of the following types of SQL statements are used:

- SQL statements that use a plug-in
- SQL statements that use the system definition scalar facility
- SQL statements that use a routine

Specification guidelines

For details about how to determine the value for this operand, see *C.6 Formulas for determining size of routine definition information buffer (pd_routine_def_cache_size)*.

Tuning the specified value

For details about how to tune the size of the routine definition information buffer, see the *HiRDB Version 9 System Operation Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `100`.

Note

If the value specified for this operand is smaller than the total of the sizes of the definition information for all plug-ins, the definition information on plug-in facilities will not be allocated in the buffer.

Effects on individual estimation formulas

If the value of the `pd_routine_def_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

20) pd_registry_cache_size = *registry-information-buffer-size*

~<unsigned integer>((0-65536)) (kilobytes)

This operand is related to plug-ins. If you use a plug-in that uses registry information, we recommend that you use this operand. We also recommend that you use this operand if you use HiRDB Text Search Plug-in.

Specify the size of the buffer (shared memory) for storing registry information (in kilobytes). When registry information is used, it is stored in the buffer. Registry information is used during the execution of an SQL statement.

Advantages

- Once registry information is stored in this buffer, it is not necessary to access the registry when the same information is used subsequently, thus reducing the number of I/O operations and CPU usage time.
- The number of communications with the dictionary server is reduced.
- Specifying this operand can improve performance when a plug-in that makes frequent use of registry information is used.

Specification guidelines

Use the following formula to determine the registry information buffer size:

$\uparrow (0.3 + a) \uparrow \times b$ (kilobytes)

a: Average registry key length (kilobytes)

The average registry key length can be determined with the following SQL statement:

```
SELECT AVG(KEY_LENGTH) FROM MASTER.SQL_REGISTRY
```

Because the result of this SQL statement is output in bytes, convert it to kilobytes.

b: Number of registry keys registered

The number of registry keys registered can be determined with the following SQL statement:

```
SELECT COUNT(*)FROM MASTER.SQL_REGISTRY
```

Tuning the specified value

For details about how to tune the size of the buffer for registry information, see the *HiRDB Version 9 System Operation Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `10`.

Effects on individual estimation formulas

If the value of the `pd_registry_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

6.2.8 Operands related to RPC trace information

21) pd_rpc_trace = Y | N

Specifies whether RPC trace information is to be collected. HiRDB maintenance information is output in the RPC trace information.

Normally, omit this operand.

`Y`: Collect RPC trace information.

`N`: Do not collect RPC trace information.

Note

Specifying `Y` for this operand degrades communication performance.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `N` is assumed.

22) pd_rpc_trace_name = "*name-of-RPC-trace-collection-file*"

~<path name of up to 254 characters>

Specifies as an `absolute` path name the file name for the RPC trace files. Three RPC trace files are created, with 1, 2, and *l* suffixed to the specified file name.

Note

Files with a maximum size of *pd_rpc_trace_size value* × 2 are created under the directory specified by this operand. Pay attention to the file size.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, the following value is assumed:

`%PDDIR%\spool\rpctr`

23) pd_rpc_trace_size = *RPC-trace-collection-file-size*

~<unsigned integer>((1024-2147483648)) (bytes)

Specifies in bytes the size of the RPC trace files.

The value specified by this operand determines the size of both RPC trace file 1 and RPC trace file 2.

Specification guidelines

An RPC trace collects in a single file in time order the information on the communications among all processes. If the volume of this information exceeds the specified value, trace data output is switched to the other file, which means that older trace information will be overwritten. When this happens, there might not be enough trace information available, making troubleshooting difficult. For this reason, specify at least `1000000` for this operand.

Note

The value specified by this operand does not apply to RPC trace file *l*, because its size is fixed at 0 bytes.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `4096` is assumed.

6.2.9 Operands related to troubleshooting information

24) pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*

~<unsigned integer>((126-16383))

A HiRDB process records the history of the executed functions and macros inside the process private memory. This history is called a *module trace*. This operand specifies the number of module trace records. The content of this history is loaded into the `core` file and is output when a process error occurs.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. If the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `126`.

Note

Process private memory of the following size is allocated to each process:

In the 32-bit mode: 64 + 48 × *value of pd_module_trace_max operand* (bytes)

In the 64-bit mode: 64 + 64 × *value of pd_module_trace_max operand* (bytes)

25) pd_module_trace_timer_level = 0 | 10 | 20

Specifies how to acquire the time to be output in module traces. The following table explains the meaning of the value specified for this operand.

Specified value	Time acquisition method
`0`	Time is output in seconds at every module trace output location.
`10`	Time is output in microseconds only at performance-critical module trace output locations, such as those before and after input/output processing, and time is output in seconds at other locations.
`20`	Time is output in microseconds at every module trace output location.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. If the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `0`.

Note

If you specify a value other than `0` for this operand, a function for acquiring time in microseconds is issued, and as a result, system performance might decline.

26) pd_pth_trace_max = *maximum-number-of-stored-communication-traces*
~<unsigned integer>((1024-8388608))

Specifies the maximum number of communication trace records to be used as troubleshooting information.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reason such as performance checking, follow the maintenance engineer's instructions.

Operand defaults

If this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. If the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `1024`.

Notes

Increasing the value of this operand increases the amount of process private memory secured by HiRDB processes.

Process private memory for communication traces is calculated based on this operand's value rounded up to the power of two. For details about memory requirements, see *Calculating Memory Requirements* in the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_pth_trace_max` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

6.2.10 Operands related to security

27) pd_audit_def_buffer_size = *security-audit-information-buffer-length*
~<unsigned integer>((1-2000000)) (kilobytes)

Specifies (in kilobytes) the buffer size (shared memory) used for storing information for the security audit facility.

Specification guidelines

Use the following formula to determine the security audit information buffer length:

$\uparrow 0.3 + a \times 0.25 \uparrow$ (kilobytes) *a*: Number of objects for the narrowing condition that was specified in the audit trail of the security audit facility

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the buffer size shown below is acquired during HiRDB startup. If this amount of memory cannot be allocated, HiRDB starts, but the security audit information buffer is not created. In this case, the `KFPD00032-W` message is issued.

$\uparrow 0.3 + \max\{(a + 100), (a \times 1.2)\} \times 0.25 \uparrow$ (kilobytes)

a: Number of objects for the filtering condition that was specified in the audit trail of the security audit facility

Notes

If the amount of memory required by this operand's specification cannot be allocated, HiRDB will not start.

Effects on individual estimation formulas

If the value of the `pd_audit_def_buffer_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

6.2.11 Operands related to Java

Java operands are specified when a Java stored procedure or Java stored function is used. For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide*.

28) pd_java_stdout_file = *"Java-virtual-machine-standard-output-and-standard-error-output-destination-file"*

~<path name>

Specifies as an absolute path name the file to which the standard output and standard error output are to be output in a Java virtual machine.

Specification guideline

Because the size of the file specified by this operand is extremely large, this operand is not normally specified. It is recommended that this operand be specified during debugging of a Java stored procedure or Java stored function. There is no limit to the size of the file that can be specified by this operand.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. If the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the Java Virtual Machine standard output and standard error output are ignored.

Note

If there are simultaneous writing attempts from multiple processes, their output contents cannot be guaranteed.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

29) pd_java_castoff = Y | N

Specifies whether to use the following events as triggers for shutting down the process at the server (single server, front-end server, dictionary server, or back-end server) that started the Java Virtual Machine:

No.	Server type	Process name	Trigger that ends process
1	Single server	`pdsds`	UAP is disconnected
2	Front-end server	`pdfes`	UAP is disconnected
3	Dictionary server	`pddic`	Transaction is completed or UAP is disconnected
4	Back-end server	`pdbes`	Transaction is completed or UAP is disconnected

`Y`: Shut down server process when trigger occurs.

`N`: Do not shut down server process when trigger occurs.

Specification guidelines

Normally, this operand is not specified. However, if you encounter the problems described below, we recommend that you specify `Y` in this operand:

- Use of the Java Virtual Machine causes the amount of memory usage to increase to the point where the available system memory becomes nearly exhausted.
- SQL code that includes numerous search conditions is executed, and even though the connection does not use the Java Virtual Machine, the maximum stack size set by the Java Virtual Machine on another connection prevents the stack from expanding, causing the server process to be aborted by a segmentation error.

For details about the Java Virtual Machine facility, see the Java Virtual Machine documentation.

Notes

On systems that run Java stored routines frequently, specifying `Y` in this operand will generate overhead for server process restarts and Java Virtual Machine startups.

Relationship to other operands

This operand is related to the `pd_process_count` operand.

6.2.12 Operands related to system log files

30) pd_log_dual = Y | N

Specifies whether dual system log files are to be used.

`Y`: Use dual system log files.

`N`: Do not use dual system log files.

Advantages

When dual system log files are used, HiRDB collects the same system log information in both files, which are called File A and File B. If a failure occurs in one of the files while the collected system log file is being loaded, the file can be loaded from the other file, resulting in higher system reliability.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

When use of dual system log files is specified, the name of the File B system log file must be specified with the `pdlogadpf` operand.

31) pd_log_remain_space_check = warn | safe

Specifies the processing to be performed by HiRDB when the available space in the system log file falls below the warning level. This is called the facility for monitoring the free area for the system log file. For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

`warn`:

When the available space in the system log file falls below the warning level, the `KFPS01162-W` message is output.

`safe`:

When the available space in the system log file falls below the warning level, scheduling of new transactions is suppressed, all transactions inside the server are forcibly terminated, and the `KFPS01160-E` message is output.

Specification guideline

We recommend that you specify `safe` because it can reduce the probability of abnormal termination of units due to system log file space shortage. However, when `safe` is specified, all transactions inside the server are forcibly terminated when a shortage occurs in the available space in the system log file. Therefore, the design of the system log file requires more accuracy. For details about system log file design, see the *HiRDB Version 9 Installation and Design Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `warn`.

32) pd_log_auto_unload_path = "*unload-log-file-output-directory*"[,"*unload-log-file-output-directory*"]...
~<path name>((1-136 characters))

Specifies as absolute path names the unload file output directories when the automatic log unloading facility is to be used for the system log. A HiRDB file system area name must be specified to create the unload log file in a HiRDB file system area. *The directories or HiRDB file system areas specified for this operand must be created before HiRDB is started.*

Additionally, in this operand, specify a different directory or HiRDB file system area for each server.

For details about the automatic log unloading facility, see the *HiRDB Version 9 System Operation Guide*.

Specification guidelines

It is important to check the available disk space before specifying a directory, so as to ensure that the created unload log file does not cause a disk space shortage.

If an unload log file cannot be created in the specified directory because of a disk space shortage, the automatic log unloading facility stops. If this is a possibility, creation of multiple directories is recommended.

Note, however, that the database recovery operation of selecting the unload files needed for recovery is simplified somewhat when only one directory is used.

Also keep in mind the following when multiple directories are created:

- It is recommended that directories be specified in different partitions to protect against disk errors.
- If the unload log file cannot be created in a single directory because of a full disk or disk error, create an unload log file under a different directory. HiRDB uses the directories specified by this operand in the order of their specification.

Operand rules

- Up to 128 directories can be specified.
- When multiple directories are specified, the same path name cannot be specified.

Notes

- The automatic log unloading facility cannot be used in the following cases:
  - `N` is specified in the `pd_log_unload_check` operand.
- If two or more directories are created for unload log files and there is no empty directory when HiRDB starts normally, the automatic log unloading facility stops.
- When a multi-HiRDB is being used, you must create a different directory for each HiRDB. Specifying the same directory for more than one HiRDB will make it impossible to determine which unload log file applies to which HiRDB.

33) pd_log_auto_unload_restart = Y | N

Specifies whether the automatic log unloading facility is to be restarted when the system log files are swapped following the issuance of a message indicating that the automatic log unloading facility stopped due to an error (message `KFPS01150-E`).

`Y`: Restart the automatic log unloading facility.

`N`: Do not restart the automatic log unloading facility.

Condition

The following two conditions must be satisfied:

- `Y` is specified in the `pd_log_unload_check` operand or this operand is omitted.
- The `pd_log_auto_unload_path` operand is specified.

Advantages

- If `Y` is specified in this operand and a shortage of disk capacity occurs due to an increase in the number of work files, or because the unload processing fails due to a temporary error such as a process creation error, HiRDB automatically restarts the unloading of system logs the next time the system log files are swapped.
- If `N` is specified in this operand and the unload log files are set to be saved while the automatic log unloading facility is stopped due to an error, you can prevent the unload log files from being saved before the `pdlogatul -t` command is executed.

Specification guidelines

Normally, specify `Y` or omit this operand.

Specify `N` if you have already set HiRDB to monitor the automatic log unloading facility termination message (`KFPS01150-E` message) and restart the facility with the `pdlogatul -b` command.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `Y` is assumed.

34) pd_log_singleoperation = Y | N

This operand is applicable when dual system log files are used; it need not be specified when dual system log files are not used.

Specifies whether single-operation mode is to be used for the system log files. Even if an error occurs in a system log file, making no dual system log files available, HiRDB (or a unit for a HiRDB/Parallel Server) can continue processing using the remaining single normal system log file without being abnormally terminated. This is called *single operation of the system log files*.

The mode in which continuation of processing is permitted only with both system log files available (normal operation mode) is called *double operation of the system log* files.

`Y`: Use single operation of the system log files.

`N`: Do not use single operation of the system log files. Both system log files must always be used.

Condition

This operand is valid only when `pd_log_dual = Y` is specified.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `N`.

35) pd_log_rerun_reserved_file_open = Y | N

Specifies whether a system log file is to be opened automatically.

If no system log file that can be overwritten is available during a unit restart, HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of system log file*.

A reserved file is used in the following cases:

- Between the time of a restart and the time when the first synchronization point dump is collected
- When none of the opened file groups can be overwritten

`Y`: Open a system log file automatically (open and use a reserved file).

`N`: Do not open a system log file automatically (do not use a reserved file).

Advantages

When `Y` is specified, the unit can be restarted as long as a reserved file is available, even though no file that can be swapped in is available during the unit restart.

However, if a file that is in unload wait status is available, the unit is stopped; once that file has been unloaded, the unit can be restarted.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `N`.

36) pd_log_rerun_swap = Y | N

Specifies whether the system log files are to be swapped during a unit restart.

`Y`: Swap the system log files.

`N`: Do not swap the system log files.

Advantage

When `Y` is specified, there can be a physical separation between the system log files used before and after a restart. Therefore, the system log file that was being used before the restart can be reused during server operation.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `N`.

37) pd_log_swap_timeout = *wait-time-for-completion-of-system-log-file-swapping*

~<unsigned integer>((1-32580)) (seconds)

Specifies in seconds the wait time during which swapping of system log files must be completed. If a swap of system log files is not completed within the specified amount of time, the unit terminates abnormally.

Specification guidelines

Normally, there is no need to specify this operand. If it takes a long time to swap system log files for a reason such as poor machine performance, you can specify this operand with a value that is greater than the default value. To detect errors or delays quickly during system log file swapping so that the unit can be terminated (such as because of a disk failure), reduce the value of this operand.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `180`.

38) pd_log_unload_check = Y | N

Specifies whether HiRDB is to check the unload status of system log files.

`Y`:

Check the unload status (normal operation).

`N`:

Do not check the unload status. A system log file is placed in swappable status, regardless of its unload status, when both of the following conditions are satisfied:

- It is in overwritable status
- It is in extraction completed status (HiRDB Datareplicator)

In this case, the system log file operation method is to release checking of unload status. For details about this operation method, see the *HiRDB Version 9 System Operation Guide*.

Advantages

Specifying `N` provides the following advantages:

- Operations are simplified because the system log file unload operation is eliminated.
- It is not necessary to provide files for storing unload files.

Specification guideline

Specify `N` if the system log file will not be needed for database recovery (in other words, if recovery from a backup collection point will be sufficient).

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `Y`.

Notes

The following points apply when `N` is specified:

- Database can be recovered only if backups have been made.
- If this option is specified when the system log file is required for database recovery, it will not be possible to recover the database.

39) pd_log_max_data_size = *log-input/output-buffer-size*

~<unsigned integer>((32000-523000)) (bytes)

Specifies in bytes the size of the buffer to be used for system log input/output operations.

Specification guideline

Specify a value that satisfies conditional expression 1 below. If `uap` is specified in the `pd_rpl_reflect_mode` operand or a recovery-unnecessary front-end server is used, specify a value that satisfies both the conditional expressions below (1 and 2). To optimize the value, use the tuning method for the specification value.

Conditional expression 1: log input/output buffer length $\geq a$

a: 72 $\times$ (*maximum number of back-end and dictionary servers that are targets for reference or update processing by a single transaction*) + 1,344

Conditional expression 2: log input/output buffer length $\geq b$

b: (*maximum number of back-end and dictionary servers that are targets for reference or update processing by a single transaction* + 1) × 128 + 64

Tuning the specified value

A value (other than the default value) might need to be specified for this operand after the following types of statistics analysis utility statistical information related to system operation have been checked:

- Number of buffer sectors waiting for input/output (`# OF BUFFER FOR WAIT I/O`)

  If the average number of buffer sectors waiting for input/output significantly exceeds 100, increase the value for this operand so that the average approaches 100.

- Number of waits caused by lack of a current buffer (`# OF WAIT THREAD`)

  If the number of waits caused by lack of a current buffer is not 0, increase the value for this operand.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `400000` (however, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `32000`).

Notes

The specification of this operand affects the response and throughput of SQL code executed by a transaction. When a small value is specified, writing to system log files occurs frequently, which might result in deterioration of performance.

Relationship to other operands

Use this operand and the `pd_log_write_buff_count` operand to determine the log I/O buffer size.

40) pd_log_write_buff_count = *log-output-buffer-sectors-count*

~<unsigned integer>((3-65000))

Specifies the number of buffer sectors to be used for system log output.

Tuning the specified value

Specify `10` (the default) for this operand initially. Thereafter, use a statistics analysis utility to obtain statistical information related to system operation to check the number of waits caused by a shortage of current buffer space (`# OF WAIT THREAD`). If the number of waits is high, specify a larger value in order to improve throughput.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `10` (however, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `3`).

Notes

If a small value is specified in this operand and the number of transactions is high, multiple transactions might be waiting for system log output, lowering system performance.

Relationship to other operands

Use this operand and the `pd_log_max_data_size` operand to determine the log output buffer sector count.

41) pd_log_rec_leng = *system-log-file-record-length*

~<unsigned integer>((1024, 2048, 4096)) (bytes)

Specifies the record length for the system log files; the specifiable values are `1024`, `2048`, and `4096`.

Specify the record length specified in the `-l` option of the `pdloginit` command for this operand.

Specification guidelines

Specify the record length based on the guidelines for designing the record length of system log files. For details about the guidelines for designing the record length of system log files, see *Record length of a system log file* in the *HiRDB Version 9 Installation and Design Guide*.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `4096`.

Notes

- If a value that is different from the record length specified by the `-l` option of the `pdloginit` command is specified for this operand, system log files cannot be opened.
- For details about how to modify the system log file record length, see the *HiRDB Version 9 System Operation Guide*.

42) pd_log_rollback_buff_count = ***rollback-log-input-buffer-sector-count***

~<unsigned integer>((0-256))

Specifies the number of buffer sectors to be used for system log input during rollback processing. When `0` is specified in this operand, HiRDB determines the number of rollback log input buffer sectors.

Specification guidelines

- we recommend that you specify `0` in this operand, in which case HiRDB will calculate an appropriate value automatically.
- If the specified value is too small, concurrent execution of rollbacks might be slowed. If the specified value is too large, the unit controller might use more shared memory than is necessary.

Tuning the specified value

Specify `0` in this operand. If specifying `0` leads to memory shortages, do not specify this operand.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is as shown below:

Number of servers in unit × 2

Effects on individual estimation formulas

If the value of the `pd_log_rollback_buff_count` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

43) pd_log_auto_expand_size = ***extension-amount-per-system-log-file-extension-trigger*****[,*****extension-limit*****]**

~<unsigned integer>((0-104857600))<<0,0>>(records)

Specify this operand when the system log file automatic extension facility is used.

Specifies the number of records to be added to the system log file for each extension trigger and an upper limit for extending file size.

If the amount to be added per trigger is omitted or `0` is specified, the system log file will not be extended automatically. If the extension limit is omitted or `0` is specified, the system log file might be extended until either the disk on which the file system area is located becomes full or the system log file capacity reaches its upper limit. When a value is specified for the amount to be added per trigger that is larger than the extension limit, the file will be extended up to the extension limit.

For details about the system log file automatic extension facility, see the *HiRDB Version 9 System Operation Guide*.

Condition

`Y` must be specified in the `pd_large_file_use` operand or specification of this operand must be omitted.

Specification guidelines

- Specify in this operand the amount to be added per trigger based on the number of records specified when the system log file was created with the `-n` option of the `pdloginit` command. Determine 10 percent of the average number of records for all system log files and specify that value.
- Normally, an extension limit is omitted.

Tuning the specified value

If the system log output volume exceeds the expanded size after automatic extension, the system log file could become full, resulting in a unitdown. In such a case, increase the specified value (number of records to be added per trigger). If extension processing requires so much time that it affects transaction performance, specify a smaller value.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition is assumed.

6.2.13 Operands related to synchronization point dump files

44) pd_spd_dual = Y | N

Specifies whether to use dual synchronization point dump files.

`Y`: Uses dual synchronization point dump files.

`N`: Does not use dual synchronization point dump files.

Advantage

When dual synchronization point dump files are used, HiRDB collects the same synchronization point dump in both dump files A and B. Even when an error occurs in one of the files when the collected synchronization point dump is being loaded, the synchronization point dump can be loaded from the other file, thus improving system reliability.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `N` is assumed.

Relationship to other operands

To use dual synchronization point dump files, specify the name of synchronization point dump file B in the `pdlogadpf` operand.

45) pd_spd_assurance_msg = Y | N

Specifies whether the `KFPS02183-I` message is to be output when a synchronization point dump is completed.

`Y`: Output the message.

`N`: Do not output the message.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `Y` is assumed.

46) pd_spd_assurance_count = *number-of-guaranteed-valid-generations*

~<unsigned integer>((1-2))

~<unsigned integer>((1-2))

Specifies the range of system log files to be saved during HiRDB operation as a protection against events such as a synchronization point dump file input error during system recovery. What is specified here is a number of synchronization point dump file generations; the specified value is referred to as the *number of guaranteed-valid generations*. The number of past generations of synchronization point dump files specified here are overwrite disabled.

Advantage

When `2` is specified as the number of guaranteed-valid generations and an error occurs in the most recent synchronization point dump file generation, the system can be recovered using the preceding synchronization point dump file generation, resulting in higher reliability.

Specification guidelines

- To improve reliability, specify `2` for the number of guaranteed valid generations. However, the number of overwrite disabled synchronization point dump files increases (to two).
- If reliability has been improved with the use of dual synchronization point dump files, we recommend that you omit this operand or specify `1` for it.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `1` is assumed.

Notes

- The minimum number of synchronization point dump files needed is *number-of-guaranteed-valid-generations* + 1.

- Specifying 2 increases the number of overwrite disabled synchronization point dump files. Moreover, the system log files that correspond to the overwrite disabled synchronization point dump files are also overwrite disabled. Consequently, specifying 2 for the number of guaranteed valid generations increases the number of overwrite disabled system log files. As a result, a shortage might occur in the number of system log files that can be swapped in. To prevent this, it might be necessary to re-evaluate the system log file capacity.

47) pd_spd_reduced_mode = *reduced-mode-operation-option*

~<unsigned integer>((0-2))

Specifies whether the reduced mode operation for synchronization point dump files is to be used.

Reduced mode operation is a facility for continuing processing if at least two synchronization point dump files can be used, even if an event such as a file error during HiRDB operation or restart reduces the number of synchronization point dump files to or below the number of required files (*number of guaranteed valid generations*# + 1).

#: Value specified for the `pd_spd_assurance_count` operand.

`0`: Do not use the reduced mode operation.

`1`: Use the reduced mode operation.

`2`: Use the reduced mode operation and issue a warning message whenever a synchronization point dump is being acquired during the reduced mode operation.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `0` is assumed.

48) pd_spd_reserved_file_auto_open = Y | N

Specifies whether a synchronization point dump file is to be opened automatically. When `Y` is specified and an error in a synchronization point dump file reduces the number of synchronization point dump files to the number of files necessary for operation (*number of guaranteed valid generations*# + 1), HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of synchronization point dump file*.

#: Value specified for the `pd_spd_assurance_count` operand.

`Y`:

Open a synchronization point dump file automatically. A reserved file is opened when the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

`N`:

Do not open a synchronization point dump file. No reserved file is opened, even though the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `N` is assumed.

Relationship to other operands

This operand has a higher priority than the `pd_spd_reduced_mode` operand.

49) pd_spd_max_data_size = *synchronization-point-dump-file-buffer-size*

~<unsigned integer>((32000-4000000)) (bytes)

Specifies in bytes the size of the buffer (shared memory) to be used for synchronization point dump file input/output operations.

The value specified here affects the number of synchronization point dump file input/output operations.

Specification guidelines

- Normally, this operand need not be specified.
- When the value of this operand is increased, the number of synchronization point dump file input/output operations is reduced.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `32768` is assumed.

50) pd_log_sdinterval = *system-log-output-volume*[,interval]

Specifies the collection interval for synchronization point dumps. This operand can be specified based on the following information:

- Volume of system log information
- Amount of time that has elapsed since the previous synchronization point dump was collected

***system-log-output-volume*: ~<unsigned integer>((100-100000)) (number of log blocks)**

Specifies an interval between synchronization point dumps in terms of the number of blocks of log information. A synchronization point dump is collected each time system log information equivalent to the number of log blocks specified here has been output.

***interval*: ~<unsigned integer>((0 or 10-1440)) (minutes)**

Specifies a synchronization point dump collection interval in terms of the number of minutes between synchronization point dumps. A synchronization point dump is collected when the time interval specified here has elapsed since the previous synchronization point dump was collected.

- If `0` is specified for the interval, HiRDB does not use a time interval for collecting synchronization point dumps.
- If no transactions execute during the time interval since the previous synchronization point dump was collected, no synchronization point dump is collected, even if the interval specified here has elapsed.

Specification guidelines

- This operand need not be specified if no specific amount of time is specified for restarting HiRDB.
- The value specified for this operand affects the amount of time required to restart HiRDB.

  Specifying a small value for this operand reduces the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps increases, online performance might deteriorate in some cases.

  Conversely, specifying a large value for this operand increases the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps decreases, online performance might improve in some cases.

Tuning the specified value

The synchronization point dump collection intervals can be checked with the statistics analysis utility; the relevant information is shown under `SYNC POINT GET INTERVAL` in the statistical information related to system operation. Use the average of the `SYNC POINT GET INTERVAL` values. If the synchronization point dump collection interval is determined to be too long, decrease the specification value; conversely, if it is determined to be too short, increase the specification value.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the following values are assumed:

- *system-log-output-volume*: `5000`
- *interval*: `60`

Note if `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, `1000` is assumed as the default value for the system log output volume.

Note

- The synchronization point dump collection interval is determined on the basis of the volume of system log information that is output. Therefore, committing data from memory to a database takes a long time during intervals that have few updating transactions. If an error occurs during such a time, it will take longer to recover the transactions that were generated during that period. If this is a possibility, also use the `interval` value to set the synchronization point dump collection interval.
- When synchronization point dumps are collected, update pages are output from the global buffer, proportionately increasing the loads on the CPU, input/output processing, and lock time. For this reason, reducing the synchronization point dump collection interval might cause processing delays.

6.2.14 Operands related to server status files

51) pd_sts_file_name_1 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

:

pd_sts_file_name_7 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

Define server status files. Although `pd_sts_file_name_2` to `7` operands can be omitted, the `pd_sts_file_name_1` operand cannot be omitted.

"*logical-file-name*" ~<identifier>((1-8 characters))

Specifies the logical file name of a status file for the front-end server.

When a command that manipulates the status file is to be executed, the logical file name defined here must be specified.

"*file-a-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File A status file as an absolute path name.

"*file-b-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File B status file as an absolute path name.

Specification guidelines

- The files specified as File A and File B must be status files created with the `pdstsinit` command. If a file that has not been created with the `pdstsinit` command is specified, a virtual status file is created.
- If an error occurs in a status file, HiRDB swaps the status files. If no spare file is available for swapping, HiRDB (or a unit for a HiRDB/Parallel Server) terminates abnormally. For this reason, defining a large number of system files improves system reliability, but at the expense of increasing the amount of disk space that is required.
- The status files specified for File A and File B must have the same record length and capacity.

Operand rules

- Up to seven instances of this operand can be specified.
- A status file has a dual structure consisting of File A and File B; both must be specified.
- Environment variables cannot be used for the absolute path names of the File A and File B file names.
- The same names cannot be specified for the logical file name, the File A file name, and the File B file name.
- HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sts01`, `C:\hirdb\sysfile` is not case sensitive, but `sts01` is case sensitive.

Notes

- When HiRDB is started normally, the primary file (the file that was the primary file when HiRDB was terminated) is inherited. However, if there is no current file that can be inherited, for example because all status files have been initialized, the first status file that was specified among those specified by the `pd_sts_file_name_1` to `7` operands becomes the current file, and the remaining files become spare files if they can be opened. Any files that cannot be opened become reserved files.
- When HiRDB is restarted, the primary file (the file that was the primary file when HiRDB was terminated) is inherited.

Using a virtual status file

Specifying a virtual status file makes it possible to add a new status file while HiRDB is running.

For example, if the number of spare files is reduced because of errors in status files, virtual status files can be converted into spare files. The procedure for converting virtual status files into spare files follows.

Procedure

1. Use the `pdstsinit` command to create a status file in a HiRDB file system area for system files.
2. Use the `pdstsopen` command to open the status file.

These operations can be performed while HiRDB is running; there is no need to stop HiRDB.

- **Advantages and disadvantages**

Defining a virtual status file reduces the amount of space required in the HiRDB file system area. However, a virtual status file cannot be added as a spare file if the HiRDB file system area for system files does not have sufficient space (sufficient space for adding the file), thus reducing system reliability.

When virtual status files are not defined, the amount of space required in the HiRDB file system area is increased. However, because a swap destination is guaranteed when a file error occurs, system reliability is increased.

- **Notes**

  When virtual status files are defined, HiRDB determines that a server status file error has occurred when the server is started. For this reason, `pd_sts_initial_error = continue` must be specified. It is also necessary before starting the unit to specify in the `pd_sts_last_active_file` operand the current file that was active during the previous operation.

6.2.15 Operands related to server status files (when an error occurs)

For details about the measures to be taken when an error occurs in a status file, see the *HiRDB Version 9 System Operation Guide*.

52) pd_sts_initial_error = stop | continue | excontinue

When a server (single server, front-end server, dictionary server, or back-end server) starts, HiRDB performs a process of identifying the current server status file. This operand specifies the action that HiRDB takes when any of the following errors is detected during this identification process.

- No real server status file is found.
- An error is detected in the server status file.

The current file identification process is applied to the server status files specified by the `pd_sts_file_name_1` to `7` operands.

`stop`:

When an error is detected in a server status file during the current file identification process, the startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started. In this case, first take a corrective action for the status file in which the error was detected, and then start the unit.

`continue` or `excontinue`:

Even when an error is detected in the server status file during the current file identification process, the startup of the server is continued if the current file is normal. However, the startup might be stopped depending on the value specified for the `pd_sts_singleoperation` operand (whether operation continues with a single status file). The following table shows the relationship to the `pd_sts_singleoperation` operand.

● **Relationship to the pd_sts_singleoperation operand**

pd_sts_singleoperation operand value	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
`continue`	When an error is detected in a server status file, HiRDB cannot identify the current file, and thus startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	The HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.
`stop` (default value)	When an error is detected in a server status file, HiRDB identifies the current file and startup of the server is continued. However, if the primary and secondary files satisfy any of the conditions listed in the table below (cases in which HiRDB cannot identify the current file), startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	If HiRDB cannot identify the current file, the HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.

- **When HiRDB cannot identify the current file**

pd_sts_initial_error operand value	File A status	File B status
`continue`	Error shutdown	Error shutdown
	Error shutdown	Open (initial state)
	Error shutdown	No real file
	Open (initial state)	Error shutdown
	Open (initial state)	No real file
	No real file	Open (initial state)
	No real file	Error shutdown
	No real file	No real file
`excontinue`	Error shutdown	Error shutdown
	Error shutdown	No real file
	No real file	Error shutdown
	No real file	No real file

Therefore, to minimize actions by the HiRDB administrator (to reduce the number of cases in which both the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands must be specified), specify the following:

- `pd_sts_initial_error = excontinue`
- `pd_sts_singleoperation = stop`

Specification guidelines

The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_initial_error operand value	
	stop	continue or excontinue
Processing by HiRDB during server startup	When an error is detected in a server status file, startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	Even when an error is detected in a server status file, the startup of the server is continued if the current file is normal.
Specification guideline	To improve system reliability, specify `stop`.	To simplify the error-handling actions during HiRDB startup, specify `continue` or `excontinue`.
Advantage	Guarantees that all server status files of the server are normal when the server starts. Therefore, if an error occurs in the current file after HiRDB has started, it can be swapped to a spare file.	Even when an error is detected in a server status file during server startup, HiRDB can start with the remaining normal files only. Therefore, HiRDB stop time can be shortened. In this case, because the number of spare files has become small, it is necessary to immediately repair the status files containing errors.
Disadvantage	Possibility increases that an error in a server status file stops the startup of HiRDB.	Because HiRDB might be running with only a small number of spare files, system reliability is low. Depending on the number of spare files available, it might not be possible to swap server status files.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `stop` is assumed.

Notes

- If both current files are abnormal, startup of the server is stopped regardless of the value specified for this operand, and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.
- Before starting HiRDB, do not initialize the current file with the `pdstsinit` command. If the current file is initialized, HiRDB cannot be restarted.
- If `excontinue` is specified for this operand and all generations of the status files are in single-operation mode and open (initial status), HiRDB does not start.

Remarks

The following figure shows the values specified for this operand, the processing performed by HiRDB, and the actions to be taken by the HiRDB administrator.

● Values specified for this operand and the processing performed by HiRDB

pd_sts_initial_error operand value	Status file error	pd_sts_singleoperation operand value	Can HiRDB identify the current file?	Specification of pd_sts_last_active_file	Does the pd_sts_last_active_file operand value match the latest file that can be opened?	Current file error	Specification of the pd_sts_last_active_side operand	Is the file specified for the pd_sts_last_active_side operand usable?	Numbers listed in the Table Processing performed by HiRDB and actions to be taken by HiRDB the administrator
`stop` (default value)	None	—	—	—	—	—	—	—	[1]
	Yes	—	—	—	—	—	—	—	[5]
`continue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
`excontinue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]

Legend:

—: Not applicable (the condition does not affect the processing performed by HiRDB)

● Processing performed by HiRDB and the actions to be taken by the HiRDB administrator

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[1]	HiRDB startup processing continues.	None
[2]	HiRDB identifies the latest current file and continues the startup processing.	Make the file that is in the error-shutdown state into a spare file.

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[3]	Using the file specified in the `pd_sts_last_active_file` operand as the current status file, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[4]	Using the files specified in the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands as the current status files, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[5]	Because `stop` is specified for the `pd_sts_initial_error` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000010` in message `KFPS01005-E`.
[6]	Because the current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000016` in message `KFPS01005-E`.
[7]	Because the normal current file identified by HiRDB does not match the file specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000017` in message `KFPS01005-E`.
[8]	Because the current file name identified by HiRDB does not match the file name specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000015` in message `KFPS01005-E`.
[9]	Because the normal current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000018` in message `KFPS01005-E`.

53) pd_sts_singleoperation = stop | continue

Specifies whether processing of server status files continues in the single-operation mode. The status file single-operation mode means that processing is to continue using only the normal file (single file) when an error occurs in the status file and a spare file is not available. For details about the status file single-operation mode, see the *HiRDB Version 9 Installation and Design Guide*.

If an error occurs in one of the current files and a spare file is available, the status file is swapped and processing continues regardless of the value specified for this operand (operation in the single-operation mode does not occur).

`stop`:

Do not permit operation in the single-operation mode. If operation in the single-operation mode is necessary, HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated. If HiRDB is abnormally terminated, allocate spare files and then restart HiRDB.

`continue`:

Enable operation in the single-operation mode. When the single-operation mode goes into effect, the message `KFPS01044-I` is output. If an error occurs in the normal file during operation in the single-operation mode or if HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated while the status file is being updated, HiRDB cannot be restarted. Therefore, when the single-operation mode goes into effect, immediately allocate spare files.

Specification guidelines

- We recommend that you specify `stop` to increase system reliability. We also recommend that you increase the number of spare files to guard against errors in the current files.
- The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_singleoperation operand value	
	stop	continue
Specification guideline	To improve system reliability, specify `stop`.	Specify `continue` if it is important not to stop HiRDB.
Advantage	When an error occurs in one of the current files and a spare file is not available, operation in the single-operation mode does not occur and HiRDB is abnormally terminated. Consequently, the possibility of losing the content of the current files is reduced.	Even when an error occurs in one of the current files and a spare file is not available, processing can be continued. Therefore, the possibility that an error in the status file stops HiRDB is reduced.
Disadvantage	Possibility that an error in the status file stops HiRDB increases. However, increasing the number of spare files can reduce this possibility.	If an error occurs in the normal status file during operation in the single-operation mode or if HiRDB is abnormally terminated during updating of the status file, the content of the current file is lost and HiRDB cannot be restarted.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `stop` is assumed.

Relationship to other operands

The combination of the values specified for the `pd_sts_singleoperation` and `pd_sts_initial_error` operands determines the processing by HiRDB when an error occurs in the status file. Therefore, the values to be specified for these operands must be determined together.

54) pd_sts_last_active_file = "*logical-file-name*"

~<identifier> ((1-8 characters))

Specifies the name of the logical file to be used as the current status file at the time of the front-end server startup. HiRDB compares the file name specified in this operand with the file name selected by HiRDB to be the current file. If the file names match, HiRDB (or a unit for a HiRDB/Parallel Server) is started; otherwise, HiRDB (or a unit for a HiRDB/Parallel Server) is not started.

Conditions

The following conditions must be satisfied:

- `continue` or `excontinue` is specified in the `pd_sts_initial_error` operand.
- It cannot be determined if the current file selected by the HiRDB system was the most recent current file during the previous session.

Specification guidelines

1. To start HiRDB immediately after initializing all status files:

   From among the operable logical files specified in the `pd_sts_file_name1` to `7` operands, specify the one that has the smallest number. Forced startup will be used in this case, regardless of the previous termination mode.

2. When both of the current files are normal:

   Specify the name of the current file.# If HiRDB cannot be started even though the name of the current file is specified, the current file might have been initialized. In this case, first initialize all status files, and then use the method in step 1 above to start HiRDB (forced startup will be used, regardless of the previous termination mode).

3. When one of the current status files has an error:

   Use the method in step 2 above, with the following operands specified:

   - `pd_syssts_singleoperation = continue`
   - `pd_sts_last_active_side`

4. When both of the current status files have errors:

   Initialize all status files, then execute the method in step 1 above (forced startup will be used, regardless of the previous termination mode).

5. When a virtual status file is specified:

   Specify the name of the current file.#

#: The names of the current files (that were active at the end of the previous operation) can be determined from the following messages:

- `KFPS01001-I`
- `KFPS01010-E`
- `KFPS01011-I`
- `KFPS01063-I`

Of the status files displayed by these messages, the one that is reported in the message that was output most recently is the current file.

55) pd_sts_last_active_side = A | B

Specify this operand if you want to start the front-end server when one of the current files is in an error state. Specify the normal status file for this operand. HiRDB compares the file specified in this operand with the file selected by HiRDB. If they match, HiRDB copies the contents of the normal status file to secondary File A and File B, then HiRDB switches the spare file in as the current file and starts the unit. If the files do not match, HiRDB (or a unit for a HiRDB/Parallel Server) is not started.

Conditions

The following operands must be specified:

- `pd_sts_initial_error = continue` or `excontinue`
- `pd_sts_last_active_file`

6.2.16 Operands related to system log file configuration

56) pdlogadfg -d sys -g *file-group-name* [ONL]

Specifies a file group for a system log file. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign system log files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies a name for the file group. All file group names must be unique within a server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 200 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 200 times.

Notes

Before adding, modifying, or deleting this operand when HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while HiRDB Datareplicator is running, its extraction process might fail.

57) pdlogadpf -d sys -g *file-group-name* -a "*system-log-file-name*" [-b "*system-log-file-name*"]

Specifies the system log files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file that comprises the file group. Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

-b "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file B when dual system log files are to be used (`pd_log_dual = Y` specified). If `pd_log_dual = Y` is not specified, the system log file name is invalid, even if it is specified.

Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\log01`, `C:\hirdb\sysfile` is not case sensitive, but `log01` is case sensitive.

Notes

Before adding, modifying, or deleting this operand when HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while HiRDB Datareplicator is running, its extraction process might fail.

6.2.17 Operands related to synchronization point dump file configuration

58) pdlogadfg -d spd -g *file-group-name* [ONL]

Specifies a file group for synchronization point dump files. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign synchronization point dump files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<<identifier>((1-8 characters))

Specifies a name for the file group. All file group names must be unique within a server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 30 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 60 times.

59) pdlogadpf -d spd -g *file-group-name* -a "*synchronization-point-dump-file-name*" [-b "*synchronization-point-dump-file-name*"]

Specifies the synchronization point dump files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name*: ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*synchronization-point-dump-file-name*": ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file that comprises the file group. Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

-b "*synchronization-point-dump-file-name*": ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file B when dual synchronization point dump files are to be used (`pd_spd_dual = Y` specified). If `pd_spd_dual = Y` is not specified, the synchronization point dump file name is invalid, even if it is specified.

Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sync01`, `C:\hirdb\sysfile` is not case sensitive, but `sync01` is case sensitive.

6.2.18 Operands related to Plug-ins

60) pdplgprm -n *plug-in-name* [-s *shared-memory-size*]

Specifies the name of a plug-in and the size of the memory to be shared by the plug-in. Omit this operand if no plug-ins are to be used or no plug-ins will run in front-end servers.

Conditions

The plug-in specified here must have been registered in HiRDB with the `pdplgrgst` command.

-n *plug-in-name*: ~<identifier>((1-30 characters))

For details about the names of plug-ins that can be specified here, see the manuals for the plug-ins.

-s *shared-memory-size*: ~<unsigned integer>((1-2000000)) <<0>> (kilobytes)

Specifies in kilobytes the size of the shared memory to be used by the plug-in. For details about the size of the shared memory to be used by the plug-in, see the manual for the applicable plug-in.

Effects on individual estimation formulas

If the value of the `pdplgprm` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula for the shared memory used by a front-end server*

7 Dictionary Server Definition

This chapter explains the operands of the dictionary server definition.

7.1 Operand formats

A dictionary server definition defines information for a dictionary server. This section explains the formats used to specify the operands of a dictionary server definition. Note that the numbers in the following table correspond to the numbers assigned to the explanations of the individual operands in *7.2 Operand explanations*.

For users who are creating HiRDB system definitions for the first time

The first step is to determine the values to be specified for the operands shown in boldface type. In principle, HiRDB can be started once the boldface operands have been specified.

No.	Format	Operand category
1	`[set pd_max_dic_process =` *maximum-number-of-activated-processes-per-dictionary-server*`]`#	Processes
2	`[set pd_process_count =` *resident-processes-count*`[,`*resident-processes-count-at-server-startup*`]]`#	
3	`[set pd_server_cleanup_interval =` *interval-for-stopping-nonresident-server-processes*`]`#	
4	`[set pd_max_ard_process =` *asynchronous-READ-process-count*`]`#	
5	`[set pd_dfw_awt_process =`*number-of-parallel-writes-for-deferred-write-processing*`]`	
6	`[set pd_work_buff_mode = each | pool]`#	Work tables
7	`[set pd_work_buff_size =` *work-table-buffer-size*`]`#	
8	`[set pd_work_buff_expand_limit =` *work-table-buffer-expansion-limit*`]`#	
9	`[set pd_spd_syncpoint_skip_limit =` *maximum-number-of-skipped-synchronization-point-dumps*`]`#	System monitoring
10	`[set pd_dfw_syncpoint_skip_limit =` *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*`]`#	
11	`[set pd_lck_pool_size =` *server-lock-pool-size*`]`#	Lock
12	`[set pd_lck_pool_partition =` *per-server-lock-pool-partition-count*`]`#	
13	`[set pd_lck_until_disconnect_cnt =` *total-number-of-tables-and-RDAREAs-to-be-locked-per-server-UNTIL-DISCONNECT-specification*`]`#	
14	`[set pd_max_open_holdable_cursors =`*maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*`]`#	
15	`[set pd_lck_hash_entry =` *lock-pool-hash-entry-count*`]`#	
16	`[set pd_dbsync_lck_release_count =` *global-buffer-lock-release-interval-during-synchronization-point-processing*`]`#	
17	`[set pd_sql_object_cache_size =` *SQL-object-buffer-size*`]`#	Buffers
18	`[set pd_dic_shmpool_size =` *dictionary-server-shared-memory-size*`]`#	Shared memory
19	`[set pd_rpc_trace = Y | N]`#	RPC trace information
20	`[set pd_rpc_trace_name = "`*name-for-RPC-trace-collection-files*`"]`#	
21	`[set pd_rpc_trace_size =` *RPC-trace-collection-file-size*`]`#	

No.	Format	Operand category
22	[set pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*]#	Troubleshooting information
23	[set pd_module_trace_timer_level = 0 | 10 | 20]#	
24	[set pd_pth_trace_max = *maximum-number-of-stored-communication-traces*]#	
25	[set pd_max_add_dbbuff_no = *maximum-global-buffers-count-for-dynamic-addition*]#	Global buffers
26	[set pd_max_add_dbbuff_shm_no = *maximum-shared-memory-segments-count-for-dynamic-addition*]#	
27	[set pd_java_stdout_file = "*Java-virtual-machine-standard-output-or-standard-error-output-destination-file*"]#	Java
28	[set pd_java_castoff = Y | N]#	
29	**[set pd_log_dual = Y | N]**#	System log files
30	[set pd_log_remain_space_check = warn | safe]#	
31	[set pd_log_auto_unload_path = "*unload-log-file-output-directory*" [,"*unload-log-file-output-directory*"]...]	
32	[set pd_log_auto_unload_restart = Y | N]#	
33	[set pd_log_singleoperation = Y | N]#	
34	[set pd_log_rerun_reserved_file_open = Y | N]#	
35	[set pd_log_rerun_swap = Y | N]#	
36	[set pd_log_swap_timeout = *wait-time-for-completion-of-system-log-file-swapping*]#	
37	[set pd_log_unload_check = Y | N]#	
38	[set pd_log_max_data_size = *log-input/output-buffer-size*]#	
39	[set pd_log_write_buff_count = *log-output-buffer-sectors-count*]#	
40	[set pd_log_rec_leng = *system-log-file–record-length*]#	
41	[set pd_log_rollback_buff_count = *rollback-log-input-buffer-sector-count*]#	
42	[set pd_log_auto_expand_size = *-extension-amount-per-system-log-file-extension-trigger[,extension-limit]*]#	
43	**[set pd_spd_dual = Y | N]**#	Synchronization point dump files
44	[set pd_spd_assurance_msg = Y | N]#	
45	[set pd_spd_assurance_count = *number-of-guaranteed-valid-generations*]#	
46	[set pd_spd_reduced_mode = *reduced-mode-operation-option*]#	
47	[set pd_spd_reserved_file_auto_open = Y | N]#	
48	[set pd_spd_max_data_size = *synchronization-point-dump-file-buffer-size*]#	
49	[set pd_log_sdinterval = *system-log-output-volume*[, *interval*]]#	
50	**set pd_sts_file_name_1 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"**	Server status files
	[set pd_sts_file_name_2 = "*logical-file-name*"	

No.	Format	Operand category
	,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_3 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_4 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_5 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_6 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
	[set pd_sts_file_name_7 = "*logical-file-name*" ,"*file-a-status-file-name*","*file-b-status-file-name*"]	
51	[set pd_sts_initial_error = stop | continue | excontinue]#	Server status files (when an error occurs)
52	[set pd_sts_singleoperation = stop | continue]#	
53	[set pd_sts_last_active_file = "*logical-file-name*"]	
54	[set pd_sts_last_active_side = A | B]	
55	**pdwork -v "*HiRDB-file-system-area-name*"[,"*HiRDB-file-system-area-name*"]...**	Work table files
56	**{{pdlogadfg -d sys -g *file-group-name* [ONL]}}**	System log file configuration
57	**{{pdlogadpf -d sys -g *file-group-name* -a "*system-log-file-name*" [-b "*system-log-file-name*"]}}**	
58	**{{pdlogadfg -d spd -g *file-group-name* [ONL]}}**	Synchronization point dump file configuration
59	**{{pdlogadpf -d spd -g *file-group-name* -a "*synchronization-point-dump-file-name*" [-b "*synchronization-point-dump-file-name*"]}}**	
60	{{[pdplgprm -n *plug-in-name* [-s *shared-memory-size*]]}}	Plug-ins

#: If this operand is omitted, the value specified in the same operand in the server common definition is used.

7.2 Operand explanations

7.2.1 Operands related to processes

1) pd_max_dic_process = *maximum-number-of-activated-processes-per-dictionary-server*

~<unsigned integer>((1-2048))

Specifies the maximum number of processes that can be activated per dictionary server. When multiple front-end servers are used, processes that exceed the value specified in `pd_max_users` can sometimes become concentrated in a single dictionary server. Even when multiple front-end servers are not used, processes that exceed the value specified in `pd_max_users` can become concentrated in a single dictionary server if the number of executions of operation commands related to RDAREAs and global buffers (`pdbufls`, `pddbls`, `pdopen`, `pdclose`, `pdhold`, and `pdrels`) exceeds the value specified in `pd_max_users`.

The `pd_max_dic_process` operand specifies the maximum number of processes that can be activated per dictionary server when that number exceeds the value of the `pd_max_users` operand.

Specification guidelines

- The value determined by the following formula indicates the maximum number of processes that might become concentrated in a single dictionary server.
  pd_max_users value × *number of front-end servers*
  Specify a value for the `pd_max_dic_process` operand by using the value determined here as the upper limit and taking the degree of process concentration in a single back-end server into consideration. Specifying an unnecessarily large value might cause a memory shortage.
- If the execution of operation commands related to RDAREAs or global buffers causes processes to become concentrated in the dictionary server, the number of operation commands to be concurrently executed is used as the maximum value.
- If the value specified is smaller than the `pd_max_users` value, the `pd_max_users` value is assumed as the default and a warning message (`KFPS01888-W`) is output.
- When more processing requests have been issued than there are dictionary server processes that can be started, delays will result.

Operand default

When this operand is omitted, the specification of the same operand in the system common definition is assumed. When the same operand is also omitted in the system common definition, the `pd_max_users` value is assumed.

Tuning the specified value

For details about how to tune the maximum number of processes that can be activated, see the *HiRDB Version 9 System Operation Guide*.

2) pd_process_count = *resident-processes-count*[,*resident-processes-count-at-server-startup*]

~<unsigned integer>((0-2048))

resident-processes-count

Specifies the number of processes that can be made resident in the dictionary server. A resident process is a process that is activated at the time the server is started.

Advantage

By activating the processes used by transactions that can be processed concurrently by the dictionary server at the time of system startup and keeping them resident, the process startup time can be reduced even when new transactions are entered. However, HiRDB startup will take longer.

Specification guidelines

- The value to be specified is determined on the basis of the process private area of the dictionary server's server process and the amount of real memory in the processor. For details about the process private area of the server process, see the *HiRDB Version 9 System Operation Guide*.
- If the `pd_max_bes_process` or `pd_max_dic_process` operand has also been specified in the case of a multi front-end server configuration, specify this operand so that the following condition is satisfied:
  Value of pd_process_count ≤ (*value of pd_max_bes_process or value of pd_max_dic_process*)

- The value specified in this operand must be no more than the dictionary server's maximum number of processes that can be activated (*value of pd_max_dic_process*$^{\#}$).

  #: If the `pd_max_dic_process` or `pd_max_bes_process` operand is omitted, the default is the value of the `pd_max_users` operand.

Tuning the specified value

For details about how to tune the resident processes count, see the *HiRDB Version 9 System Operation Guide*.

Notes

- Because the number of resident processes has a direct effect on the availability of memory space and on the CPU, specifying an unnecessarily large number might prevent HiRDB from starting or might degrade the server machine's processing performance.
- If more processes than the resident process count are needed, additional processes are dynamically started, up to the maximum processes count allowed. However, depending on the value specified for the `pd_max_server_process` operand, it might not be possible to start all of the processes indicated by the maximum processes count.

Operand default

When this operand is omitted (or `0` is specified), the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the following value is assumed:

↑ *maximum number of processes that can be activated* ÷ 2 ↑

resident-processes-count-at-server-startup

Specifies the number of processes that can be made resident during HiRDB startup.

It is common for resident processes to be activated during HiRDB startup. If there are many resident processes, the amount of time required for HiRDB startup increases proportionately. For example, it takes approximately 1 second to activate a process on a server machine with a 100-MIPS performance rating.

The differences in processing that result depending on whether a resident processes count at server startup is specified are as follows:

- When there is no specification of a resident processes count at server startup (when, for example, `pd_process_count = 500` is specified)

  All 500 resident processes are activated during HiRDB startup, and HiRDB will not start until they are all activated. In this example, it would take a 100-MIPS server machine about 500 seconds to activate all the resident processes during HiRDB startup.
- When a resident processes count at server startup is specified (when, for example, `pd_process_count = 500, 50` is specified)

  Some of the resident processes (50 in this case) are activated during HiRDB startup, and the remaining resident processes are activated after HiRDB startup. HiRDB can be started as long as the specified number of resident processes is activated. In this example, it would take a 100-MIPS server machine about 50 seconds to activate 50 resident processes during HiRDB startup. The remaining 450 resident processes (in this example) would be activated after HiRDB startup.

Advantage

The amount of time required for HiRDB startup is reduced. Use this option when you want to reduce the HiRDB startup time as much as possible, such as when you are using the system switchover facility.

Specification guideline

Specify a value equal to the number of processes that will be required immediately after HiRDB startup is completed.

Notes

When you specify a resident processes count at server startup, recheck the value in the `PDCWAITTIME` operand of the client environment definition.

If more UAPs than the resident processes count at server startup are to be executed immediately following HiRDB startup, transaction processing might not be performed until after the remaining resident processes have been activated. Therefore, if the value specified in the `PDCWAITTIME` operand of the client environment definition is small, it might not be possible to process some UAPs due to timeouts. For details about the `PDCWAITTIME` operand, see the *HiRDB Version 9 UAP Development Guide*.

3) pd_server_cleanup_interval = *interval-for-stopping-nonresident-server-processes*

~<unsigned integer> ((0-1440)) (minutes)

Specifies in minutes the interval for checking for nonresident server processes in HiRDB that are to be stopped. The facility for stopping nonresident server processes is applied when the number of executing server processes exceeds the number of processes that can be made resident (value specified by the `pd_process_count` operand). The number of server processes that the facility stops is computed automatically by HiRDB.

Advantage

This facility improves the utilization rates of the memory and of process resources because it increases the number of nonresident processes that can be reused when the workload (number of server processes that are executing) peaks.

Specification guidelines

- For example, if there is only one hour a day during which peak workload occurs and the intervals between peaks within that hour are approximately two minutes apart, specify `2` for this operand.
- This facility does not have any effect if the number of server processes that execute concurrently during peak periods is always fewer than the number of resident processes. In this case, omit this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Tuning the specified value

Collect statistical information on system operations for each server for one week. Determine the workload peaks from the number of server processes being serviced (`# OF PROCESSES ON SERVICE`). If that peak value exceeds the currently set number of processes that can be made resident (value specified by the `pd_process_count` operand), determine the interval between individual peaks and set that number of minutes.

However, if there are ample resources, such as memory and CPU, in the server machine, adding the shortfall in the number of processes to the number of resident processes (in other words, increasing the value of the `pd_process_count` operand) is more effective in improving performance than specifying the `pd_server_cleanup_interval` operand.

Note

When this operand is omitted or `0` is specified, the system checks every 10 seconds for nonresident server processes that are waiting to be serviced and stops any such processes that are found.

4) pd_max_ard_process = *asynchronous-READ-process-count*

~<unsigned integer>((0-256))

Specify this operand if you use the asynchronous READ facility. For this operand, specify the number of processes necessary for asynchronous READ operations. For a HiRDB/Parallel Server, this operand specifies the number of processes per server (back-end server or dictionary server). For details about the asynchronous READ facility, see the *HiRDB Version 9 Installation and Design Guide*.

Condition

A value of `1` or greater must be specified for the `-m` option of the `pdbuffer` operand.

Specification guidelines

- Specify `0` or `1`. However, if a value between 2 and 256 is specified for the `-m` option of the `pdbuffer` operand, specify the same value as the `-m` option value. If a value greater than 256 is specified for the `-m` option, specify the same value as the number of disk devices that store RDAREAs and system files (the number of such disk devices per server for a HiRDB/Parallel Server) or `256`.
- Increasing the value of this operand can shorten the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. Decreasing the value of this operand might increase the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. This is because asynchronous READ processes might have to wait for processing completion.
- Because a number of processes equaling *value of this operand* × *server count* are started, determine a value for this operand by taking resources (shared memory and message queue) into consideration. For details about estimating shared memory and message queue sizes, see the *HiRDB Version 9 Installation and Design Guide*.

Tuning the specified value

For details about how to tune the specification value (number of asynchronous READ processes), see the *HiRDB Version 9 System Operation Guide*.

Operand defaults

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Relationship to other operands

If you change the value of this operand, re-evaluate the value of the `pd_max_server_process` operand.

Operand rule

If you specify `0` for this operand, the asynchronous READ facility is not used.

Effects on individual estimation formulas

If the value of the `pd_max_ard_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Parallel Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

5) pd_dfw_awt_process = *number-of-parallel-writes-for-deferred-write-processing*

~<unsigned integer>((2-255))

Specify this operand when you use the *facility for parallel writes in deferred write processing* for all buffer pools. Specify for this operand the number of processes to be processed in parallel. Increasing the number of processes can shorten the write processing time. For details about the facility *for parallel writes in deferred write processing*, see the *HiRDB Version 9 Installation and Design Guide*.

Specification guidelines

Specify `2`, which is the smallest value that enables the facility for parallel writes in deferred write processing. Furthermore, to determine the value for this operand, see *Tuning deferred write processing* in the *HiRDB Version 9 System Operation Guide*.

Note

Specifying the facility for parallel writes in deferred write processing increases the number of processes, and consequently raises the CPU usage rate.

Effects on individual estimation formulas

If the value of the `pd_dfw_awt_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 4* under *Formulas for the size of the shared memory used by a dictionary server*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

7.2.2 Operands related to work tables

6) pd_work_buff_mode = each | pool

Specifies the method of allocating buffers when HiRDB creates tables.

`each`: Allocate a buffer for each work table.

`pool`: Allocate a buffer pool for each server process.

Specification guidelines

- Normally, `pool` is specified. `pool` is the appropriate specification when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.
- When the size of the process private area that can be used for work table buffers is predetermined, specify `pool`. When `pool` is specified, HiRDB efficiently allocates work table buffers to work tables.

  In such a case, the process private area is occupied on the basis of the value specified in `pd_work_buff_size`, and input/output operations on work tables are buffered in that pool. Therefore, the process private memory is occupied only to the extent of the value specified in `pd_work_buff_size`.

Notes

- If `each` is specified for this operand, the amount of memory used might increase.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `pool` is assumed (if `v6compatible` is specified in the `pd_sysdef_default_option` operand, `each` is assumed).

Effects on individual estimation formulas

If the value of the `pd_work_buff_mode` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Considerations when migrating to 64-bit mode*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

7) pd_work_buff_size = *work-table-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

Specifies in kilobytes the size of buffers for work tables to be created by HiRDB.

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
Advantage	A large work table buffer size reduces the number of I/O operations associated with data manipulation, which means that the execution time of SQL statements that use work tables is also reduced. However, because each server's process private memory is used, you must also take into account the overall size of the system memory (real memory and virtual memory) when you specify this option. If `pd_work_buff_mode = each` is specified, the memory size to be allocated is *value of pd_work_buff_size* × *required number of work tables*. Therefore, specifying an unnecessarily large value might cause a virtual memory shortage for other processes.	
Application criterion	Specify `pd_work_buff_mode = pool` when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.	
Specification guidelines	• Specify the size of the buffer to be allocated for one work table. • If a value greater than the work table memory capacity is specified for the work table buffer size, input/output to the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the work table memory capacity: *Work table memory capacity = Applicable work table size*# ÷ 2	• Specify the size of the buffer pool to be allocated for the entire server process. • Specify a value between `4352` and `5120` when a large volume of data is to be retrieved or when manipulations such as join, `ORDER BY`, and `GROUP BY` are to be performed. Specifying such a value increases the unit of sorting input/output, thus reducing the sort time. • If a large value is specified for the work table buffer size, with the total work table memory capacity for each SQL statement as the upper limit, input/output operations on the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the total work table memory capacity for each SQL statement:

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
		Total work table memory capacity per SQL statement = $a \times b + c \times d$
Notes	• When multiple users execute processes concurrently or when an SQL statement that uses multiple work tables is executed, a buffer of the specified size is allocated for each work table. Consequently, specifying a large value might result in a memory shortage. • If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.	• If the value specified for the work table buffer size is smaller than the number of work tables to be used by each SQL statement, the processing time might become longer than when `each` is specified. Specifically, specify a value that is at least equal to *maximum number of work tables for each SQL statement* × 128. The following formula can be used to determine the maximum number of work tables for each SQL statement: *Maximum number of work tables for each SQL statement* = $b + d$ • If the specified buffer size is too large to be allocated in the system, the allocation of process private memory fails and the server cannot start up.
Operand rule	Specify a multiple of 128. If the value is not a multiple of 128, it is rounded up to the next multiple of 128.	• Specify a multiple of 128. If the value is not a multiple of 128, it is rounded up to the next multiple of 128. • Specify at least 384. If a value that is smaller than 384 is specified, it is rounded up to 384.
Default value	If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `512` is assumed.	If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the following value is assumed: • 32-bit mode: `1024` • 64-bit mode: `5120`

a:

$\uparrow$ {Capacity of work table (for storing column information)[#] (kilobytes) ÷ 2} ÷ 128 $\uparrow$ × 128

b:

Maximum number of work tables (for storing column information)[#]

c:

$\uparrow$ {Capacity of work table (for storing positional information)[#] (kilobytes) ÷ 2} ÷ 128 $\uparrow$ × 128

d:

Maximum number of work tables (for storing positional information)[#]

#: For details about how to determine these values, see the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_work_buff_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

8) pd_work_buff_expand_limit = *work-table-buffer-expansion-limit*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

The size of the work table buffer to be created by HiRDB is specified by the `pd_work_buff_size` operand. Specify the `pd_work_buff_expand_limit` operand if you want to automatically expand a work table buffer when the space in this buffer becomes insufficient. The work table buffer is expanded up to the size specified by this operand.

For example, when the following values are specified for the operands, a 1,024-kilobyte work table buffer is normally allocated. When this size becomes insufficient, the work table buffer is expanded up to 2,048 kilobytes.

- `pd_work_buff_size` = `1024`
- `pd_work_buff_expand_limit` = `2048`

HiRDB expands a work table buffer in the following cases:

- The necessary work table buffer cannot be allocated when hash execution is applied to an execution method that uses hash join or subquery hash as the joining method.
- A 128-kilobyte work table buffer allocated to each work table becomes insufficient when multiple work tables are concurrently used.

Condition

The `pd_work_buff_mode` operand must be omitted or `pool` must be specified for it.

Advantage

You can prevent a work table buffer shortage (too small a value specified for the `pd_work_buff_size` operand) from causing UAP errors.

Notes

- A work table buffer is not expanded when either of the following conditions is satisfied:
  - The `pd_work_buff_expand_limit` operand is not specified.
  - `pd_work_buff_expand_limit` operand value ≤ `pd_work_buff_size` operand value
- If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.

Operand rule

Specify a multiple of 128. If a value other than a multiple of 128 is specified, it is automatically rounded up to a multiple of 128.

Relationship to other operands

When a work table buffer is expanded for the first time in a dictionary server process, the `KFPH29008-I` message is output. Note that you can use the `pd_work_table_option` operand to suppress this message output.

Note

After a work table buffer has been expanded, when the number of work tables being used by the applicable server process goes to zero, the expanded work table buffer is released. The number of work tables being used can go to zero in the following cases:

- All cursors that were being used are closed. (In this case, the number of work tables being used might not go to zero.)
- A transaction is normally terminated or cancelled when a holdable cursor is not being used.
- A UAP is disconnected from HiRDB when a holdable cursor is being used.

Remarks

Hash join, subquery hash execution is applied in the following cases:

- *Application of optimizing mode 2 based on cost* and *hash join, subquery hash execution* are specified in the `pd_additional_optimize_level` operand, the `PDADDITIONALOPTLVL` operand of the client environment definition, or the `ADD OPTIMIZE LEVEL` operand of the SQL compile option.
- `HASH` is specified for the SQL optimization specification of the joining method inside an SQL statement.
- `HASH` is specified for the SQL optimization specification of the subquery execution method inside an SQL statement.

Effects on individual estimation formulas

If the value of the `pd_work_buff_expand_limit` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

7.2.3 Operands related to system monitoring

9) pd_spd_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps*

~<unsigned integer>((0, 2-100000))

If an event such as an endless loop occurs in a UAP, acquisition of synchronization point dumps might be skipped. If several synchronization point dumps are not acquired (instead, they are skipped), there will be an increase in the number of overwrite-disabled system log files, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

If HiRDB is forcibly terminated or terminates abnormally when the number of system log files that cannot be overwritten has reached one-half or more of all system log files, a shortage of system log files occurs during rollback processing when HiRDB is restarted. In this case, HiRDB cannot be restarted unless new system log files are added. Any such restart processing will take a longer time than normal.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction). When the number of skipped synchronization point dumps reaches the specified operand value, the target transaction is cancelled forcibly and rolled back. This is called the *skipped effective synchronization point dump monitoring facility*.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

Advantage

Specifying this operand provides a means of dealing with endless loops in UAPs.

Specification guidelines

- Normally, specify `0` for this operand. When `0` is specified, HiRDB computes the upper limit for the skip count. If specifying `0` causes a problem or the `KFPS02101-I` message is issued, change the value of this operand. For guidelines on the value to specify, see the *HiRDB Version 9 System Operation Guide*.
- If the specified value is too large, all system log files might be placed in overwrite disabled status. If this happens, HiRDB terminates abnormally.
- If the specified value is too small, the number of transactions that are forcibly rolled back might increase.
- Specify this value taking into account the value of `pd_log_sdinterval` and the number of times synchronization point dumps are acquired when the transaction with the longest execution time and largest log output volume is processed.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the skipped effective synchronization point dump monitoring facility is not used.

Notes

- The monitoring provided for by the specification of this operand begins the first time a synchronization point dump is collected after HiRDB is started (including a restart).
- When this operand is specified, UAPs that are being executed in the no-log mode are also monitored. If the processing of a UAP being executed in the no-log mode is cancelled, the database cannot be automatically recovered, and thus RDAREAs are placed in the error shutdown state. Therefore, when specifying the upper limit, account also for the size of the system logs that are output from other transactions inside the server during the processing of UAPs that are executed in the no-log mode.
- If a transaction is delayed due to a high workload, the skip count increases because a synchronization point dump cannot be acquired until the delayed transaction is completed.
- If the skip count exceeds the upper limit, any process executing a transaction is forcibly terminated. In this case, the rollback logs, which are the logs output by these transactions, are output after the forced termination.
- The `pdload`, `pdmod`, `pdrorg`, `pdexp`, `pddbst`, `pdgetcst`, `pdrbal`, `pdvrup`, `pdmemdb`, and `pdextfunc` commands are not monitored by this facility.

10) pd_dfw_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*

~<unsigned integer>((0-100000))

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing is completed, acquisition of the synchronization point dump is skipped. This is because acquisition of the synchronization point dump is delayed by the deferred write processing, and the number of update buffers output by the synchronization point exceeds the number of update buffers that can be output within the synchronization point dump acquisition interval.

If more than one synchronization point dump is skipped, there will be an increase in the number of system log files that cannot be overwritten, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction) due to deferred write processing.

When the number of skipped synchronization point dumps due to deferred write processing reaches the specified operand value, HiRDB determines the maximum number of update buffers in such a manner that acquisition of a synchronization point dump can be completed within the synchronization point dump acquisition interval. If the maximum number of update buffers is exceeded, HiRDB then outputs the oldest update buffer and limits the total size of update buffers at a synchronization point. This is called the *update buffer size restriction facility*.

Advantage

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing has terminated, the unit terminates abnormally. This operand enables such abnormal termination of the unit to be avoided.

Specification guidelines

Normally, you will omit this operand. If you wish to prevent unit abnormal termination caused by the occurrence of a synchronization point before termination of deferred write processing within the synchronization point dump acquisition interval, specify 1.

If an acceptable number of times synchronization point dumps can be skipped can be determined in advance, such as from the size of the log information, specify that value.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `0` is assumed.

Operand rules

If `0` is specified in this operand, HiRDB does not use the update buffer size restriction facility.

Notes

The following notes explain the period of effectiveness of the update buffer size restriction facility. The period of effectiveness means the interval between issuance of the `KFPH23035-I` message and issuance of the `KFPH23036-I` message.

- If the number of update buffers exceeds the maximum number of update buffers determined by HiRDB, update buffers are output after update processing has executed; this degrades the update processing throughput. You can use the following formula to obtain the maximum number of update buffers determined by HiRDB:

(*synchronization point dump interval* ÷ *unit value of WRITE*#)
× (1 - (*amount of log information from the previous synchronization point dump to the pre-synchronization point*)
× (*number of buffer sectors in buffer pool* ÷ *total number of buffer sectors in buffer pool that was updated at the synchronization point*)

  #: For details about the unit value of `WRITE`, see the *HiRDB Version 9 System Operation Guide*.

- If the deferred write trigger is specified in the `pd_dbbuff_rate_updpage` or `pdbuffer -y` operand, and each operand value becomes greater than the maximum number of update buffers determined by HiRDB, the maximum number of update buffers determined by HiRDB is changed to the number of update buffers that triggers deferred write processing.

  The value of the `pdbuffer -w` operand is adjusted automatically so that up to the maximum number of update buffers is output for each buffer.

- A skipped synchronization point dump is detected during update buffer output processing at a synchronization point. Therefore, the update buffer size restriction facility might be enabled after a synchronization point dump is skipped and an error message is displayed.

- Normally, when the parallel writes facility is used, there is one output request per synchronization point for each parallel WRITE process for deferred write processing during synchronization point processing. However, if the update buffer restriction facility is used, there will be more than one output request in order to facilitate early detection of skipped synchronization point dumps.

7.2.4 Operands related to lock

11) pd_lck_pool_size = ***server-lock-pool-size***

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-2000000))**
- 64-bit mode: **((1-2000000000))**

Specifies in kilobytes the size of the shared memory area to be used by the dictionary server for locking (lock pool).

The area specified by this operand is used for tables for managing locked resources. Note the following relationship between the number of tables for managing locked resources, the number of lock requests, and the lock pool size:

Number of tables for managing locked resources = number of lock requests = size of lock pool × *coefficient*

Specification guidelines

- In the 32-bit mode, the amount of lock pool space required for six lock requests is 1 kilobyte.
- In the 64-bit mode, the amount of lock pool space required for four lock requests is 1 kilobyte.
- The following formulas can be used to determine the value to be specified for this operand:

HiRDB type	Formula (kilobytes)
HiRDB/Parallel Server (32-bit mode)	↑↑*b* ÷ *value of pd_lck_pool_partition* ↑ ÷ 6 ↑ × *value of pd_lck_pool_partition*
HiRDB/Parallel Server (64-bit mode)	↑↑*b* ÷ *value of pd_lck_pool_partition* ↑ ÷ 4 ↑ × *value of pd_lck_pool_partition*

b: Total number of transaction lock requests to be executed concurrently by the dictionary server. The number of lock requests depends on the SQL. For details about how to determine the total number of lock requests, see *D. Determining the Number of Locked Resources*.

Note:

When you execute `DROP TABLE` or `DROP SCHEMA` in a definition SQL, it is especially important to have already determined in advance an appropriate value for this operand.

Tuning the specified value

See the usage rate for the locked resources management table (`% OF USE LOCK TABLE`) displayed for the dictionary server in the statistical information on system operation by the statistics analysis utility. If the maximum usage rate equals or exceeds 80%, it is recommended that the operand's value be increased in preparation for future database expansion. If the maximum usage rate does not exceed 10%, it is recommended that the operand's value be decreased to conserve shared memory space.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the value shown below:

- 32-bit mode: `16000`
- 64-bit mode: `32000`

However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `1024`.

Note

- If the value specified for this operand is too small, an SQL statement might return an error.
- Do not specify a larger value than necessary in this operand. A large value will increase the size of the shared memory used by HiRDB, which might cause a shortage of shared memory and prevent HiRDB from starting.
- If you do an all-item search of a table that contains many rows while locking is in units of rows, this operand's value will need to be increased to reflect the large number of items, which will increase the amount of memory required. Instead, consider making the following adjustments in the UAP:
    - Acquire locks in table units.

- If you can use the unlocked search facility, perform searches unlocked.
- Narrow the search conditions, and divide the processing into several transactions.

Relationship to other operands

This operand is related to the `pd_lck_pool_partition` operand.

12) pd_lck_pool_partition = *per-server-lock-pool-partition-count*

~<unsigned integer>((1-5000))

Specifies the number of lock pool partitions to be used in locking by the dictionary server when distributing lock processing.

For details about distributing lock processing, see the *HiRDB Version 9 System Operation Guide*.

Tuning the specified value

For details about how to tune the number of lock pool partitions, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `1`.

Notes

- When the value set in this operand is too large, the size of the shared memory assigned to each lock pool partition becomes smaller, which might result in not enough lock pool partition capacity, causing SQL to return an error. Lock processing also takes time, so system performance declines. If this becomes an issue, specify a smaller value in this operand.
- The lock pool size must be at least 1 kilobyte. If a value larger than the value of `pd_lck_pool_size` is specified, the `KFPS00421-W` message will be issued and the value of `pd_lck_pool_size` will be assumed for this operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_pool_size`
- `pd_lck_deadlock_check_interval`

Effects on individual estimation formulas

If the value of the `pd_lck_pool_partition` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

13) pd_lck_until_disconnect_cnt = *total-number-of-tables-and-RDAREAs-to-be-locked-until-disconnect-specification*

~<unsigned integer>((0-140000))

Specifies the number of resources to be locked for the tables and RDAREAs that are to be held across transactions. Based on the value specified for this operand, blocks for which lock with `UNTIL DISCONNECT` is specified for the tables and RDAREAs are allocated in the shared memory.

Specification guidelines

Normally, this operand need not be specified. Specification of a value other than the default value might be necessary in the following cases:

- When the number of utilities to be executed concurrently increases
- When a holdable cursor is used

For details about how to estimate the specification value for this operand, see *C.5 Formula for determining total number of tables and RDAREAs per server locked with UNTIL DISCONNECT specified (pd_lck_until_disconnect_cnt)*.

Tuning the specified value

If the value specified for this operand is small, a transaction might roll back or a utility might terminate abnormally with return code `8`. In such cases, the message `KFPA11914-E` or `KFPH28001-E` is output. If this occurs, increase the value of this operand.

When the value of this operand is increased, the amount of required memory space increases proportionately. The required memory size can be expressed as follows: *value of this operand* × 48 (64 in the 64-bit mode) bytes.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `256`.

Effects on individual estimation formulas

If the value of the `pd_lck_until_disconnect_cnt` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the number of records in a synchronization point dump file*
- *Formula 2* under *Formulas for the size of the shared memory used by a dictionary server*

14) pd_max_open_holdable_cursors = *maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*

~<unsigned integer>((16-1024))

When you use holdable cursors for a table for which a `LOCK` statement with `UNTIL DISCONNECT` specification is not executed, this operand specifies the maximum number of holdable cursors that can be concurrently open for each transaction.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `16`.

Note

Specifying a value other than the default value for this operand increases the amount of shared memory used.

Relationship to other operands

The values specified for this operand and the following operands are used for computing the shared memory size for lock servers. For a 32-bit mode HiRDB system, if the values specified for these operands are too large, the shared memory size of the lock servers exceeds 2 GB, and as a result, HiRDB might not start. Therefore, adjust the values specified for these operands so that the shared memory size of the lock servers does not exceed 2 GB.

- `pd_max_access_tables`
- `pd_max_users`
- `pd_max_dic_process`
- `pd_lck_hash_entry`
- `pd_lck_pool_size`

For details about shared memory, see the *HiRDB Version 9 Installation and Design Guide*.

15) pd_lck_hash_entry = *lock-pool-hash-entry-count*

~<unsigned integer> ((0-2147483647))

Specifies the number of hash table entries to be used in the lock pool. According to the value specified here, HiRDB allocates a lock pool in the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand.

Consider specifying a value for this operand in the following cases:

- If you do not wish to change the shared memory size if possible when upgrading to version 06-02 or later, specify `11261`. In this case, the same number of hash entries is allocated as in the earlier version, the hash table size inside the lock pool remains the same as before.

- It is possible to improve performance by specifying in this operand a value greater than the recommended value shown below. However, specifying a value greater than variable *a* (also shown below) will not improve performance over the case in which *a* is specified.

  The recommended value is as follows:

  Recommended value = Largest prime number not exceeding MAX($\uparrow a \div 10 \uparrow$, 11,261)

Variable	Formula for computing the variable	
a	$(b + 3) \times 10 +$ *pd_lck_pool_size value* $\times$ *c*	
b	If `pd_max_users` value > `pd_max_dic_process` value	*pd_max_users value*
	If `pd_max_users` value $\leq$ `pd_max_dic_process` value	*pd_max_dic_processs value*
c	6 for the 32-bit mode; 4 for the 64-bit mode	

Operand rules

- If this operand and the `pd_lck_hash_entry` operand of the server common definition are both omitted or `0` is specified in this operand, HiRDB calculates a recommended value for the server. (However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default for this operand is `11261`.)
- When a value that is neither `0` nor a prime number is specified in this operand, HiRDB assumes that the specification is the largest prime number that does not exceed the specified value.

Note

If the value specified in this operand is too small, there might be an insufficient number of hash entries, and performance might deteriorate. If this operand is omitted, there will never be a shortage of hash entries and performance will not deteriorate due to an insufficient number of hash entries.

16) pd_dbsync_lck_release_count = ***global-buffer-lock-release-interval-during-synchronization-point-processing***

~<unsigned integer>((0, 100-1073741824))

Specifies an interval for unlocking global buffers, when global buffer locking occurs during synchronization point processing.

During synchronization point processing, search processing occurs on the buffers (update buffers) that must be applied to the disk. Normally, global buffers are unlocked at a specific interval during search processing on the update buffers.

For example, if 100 is specified in this operand, a global buffer is unlocked once when search processing on 100 sectors (global buffer sectors) is completed. After that, the global buffer is locked again and search processing is resumed. In this example, unlocking occurs once every 100 sectors.

Advantages

By specifying this operand, you can adjust the global buffer lock time during synchronization point processing. When a small value is specified in this operand, the global buffer lock time becomes short and transaction performance might improve during synchronization point processing.

To obtain the global buffer pool lock time, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Buffer pool lock time during synchronization point processing (SYNCL)`.

Specification guidelines

Normally, there is no need to specify this operand. Consider specifying this operand when both the following conditions apply:

- Transaction performance drops during synchronization point processing.
- A large number of buffer sectors is specified in the `-n` option of the `pdbuffer` operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. If the same operand is also omitted in the server common definition, the default is `10000`.

Operand rules

- If the specified value is in the range 1 to 99, 100 is set automatically.

- If 0 is specified, global buffers are locked until update buffer search processing is completed.

Notes

If a small value is specified in this operand, the update buffer search time increases due to interrupts from other transactions and CPU usage rises. The global buffers updated during that time are also output during synchronization point processing. Therefore, the number of update buffers to be output during synchronization point processing increases. To obtain the number of update buffers to be output during synchronization point processing, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Number of synchronization point output pages (SYNCW)`. As you increase the value of this operand, the lock time for a global buffer to determine the buffer to be output at a synchronization point will increase. For this reason, lock contention grows substantially during synchronization point processing, which might affect transaction performance.

7.2.5 Operands related to buffers

17) pd_sql_object_cache_size = *SQL-object-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((22-256000))**
- 64-bit mode: **((22-2000000))**

Specifies in kilobytes the size of the buffer area (shared memory) in which SQL objects are to be placed.

Specification guidelines

- SQL objects are saved in a buffer until the user's transaction has terminated. The buffer must be large enough to store the SQL objects of all transactions that will be executed concurrently.
- If the SQL object buffer hit rate is low, the performance might decreases due to the overhead of SQL statement analysis processing.
- SQL analysis can be reduced by saving the SQL objects of static SQLs in the buffer after transaction termination (until the buffer runs out of space) and sharing them among multiple users who execute the same UAP. To effectively utilize the buffer, allocate it so that the SQL objects of frequently-used UAPs are resident in the buffer.
- To estimate the buffer size, first determine the buffer size needed for UAP execution from the length of the SQL objects from the SQL statements to be issued by the UAP. Then, compute the buffer size by considering the number of UAPs that will be executed concurrently and the number of concurrently executing users.
- For details about how to estimate the length of the SQL object from a single SQL statement, see *C.2 Formulas for determining size of SQL object buffer (pd_sql_object_cache_size).*

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition and the system common definition, the following value is assumed:

(*value of pd_max_users* + 3) × 22

Tuning the specified value

For details about how to tune the SQL object buffer size, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_sql_object_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for the size of the shared memory used by a dictionary server*

7.2.6 Operands related to shared memory

18) pd_dic_shmpool_size = *dictionary-server-shared-memory-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-200000))**
- 64-bit mode: **((1-4000000000))**

Specifies the size of the area (in kilobytes) to be used by a dictionary server as part of the shared memory for the unit controller.

Specification guideline

Normally, omit this operand. If this operand is omitted, HiRDB computes a value for it (however, if `v6compatible` is specified for the `pd_sysdef_default_option` operand, `1024` is assumed). HiRDB computes this value based on the values for the related definitions. Therefore, specify the appropriate values for the related definitions. For details about the related definitions, see *Formulas for the size of the shared memory used by a dictionary server* in the *HiRDB Version 9 Installation and Design Guide*.

If the values of any of the operands of the variables used in the estimation formula are changed, HiRDB automatically re-calculates this operand value.

Tuning the specified value

If any of the following messages is output, increase the specification value of this operand.

- `KFPA20003-E`
- `KFPD00005-E`
- `KFPD00012-E`
- `KFPD00021-E`
- `KFPH20003-E`

Notes

If this operand is omitted and the appropriate values are not specified in the related definitions, the value obtained by HiRDB might be too large, resulting in the allocation of an unnecessarily large amount of shared memory, or too small, resulting in one or more of the following problems:

- A unit does not start.
- A UAP or utility does not execute.

If the value specified for this operand is unnecessarily large or unnecessarily small, the same problems also occur.

7.2.7 Operands related to the RPC trace information

19) pd_rpc_trace = Y | N

Specifies whether RPC trace information is to be collected. HiRDB maintenance information is output in the RPC trace information.

Normally, omit this operand.

`Y`: Collect RPC trace information.

`N`: Do not collect RPC trace information.

Note

Specifying `Y` for this operand degrades communication performance.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `N` is assumed.

20) pd_rpc_trace_name = "*name-of-RPC-trace-collection-file*"

~<path name of up to 254 characters>

Specifies an `absolute` path name for the file name for the RPC trace files. Three RPC trace files are created, with 1, 2, and *l* suffixed to the specified file name.

Note

Files with a maximum size of *pd_rpc_trace_size value* × 2 are created under the directory specified by this operand. Pay attention to the file size.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, the following value is assumed:

`%PDDIR%\spool\rpctr`

21) pd_rpc_trace_size = *RPC-trace-collection-file-size*

~<unsigned integer> ((1024-2147483648)) (bytes)

Specifies in bytes the size of the RPC trace files.

The value specified by this operand determines the size of both RPC trace file 1 and RPC trace file 2.

Specification guidelines

An RPC trace collects in a single file in time order the information on the communications among all processes. If the volume of this information exceeds the specified value, trace data output is switched to the other file, which means that older trace information will be overwritten. When this happens, there might not be enough trace information available, making troubleshooting difficult. For this reason, specify at least `1000000` for this operand.

Note

The value specified by this operand does not apply to RPC trace file *l*, because its size is fixed at 0 bytes.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `4096` is assumed.

7.2.8 Operands related to troubleshooting information

22) pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*

~<unsigned integer>((126-16383))

A HiRDB process records the history of the executed functions and macros inside the process private memory. This history is called a *module trace*. This operand specifies the number of module trace records. The content of this history is loaded into the `core` file and is output when a process error occurs.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. If the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `126`.

Note

Process private memory of the following size is allocated to each process:

In the 32-bit mode: 64 + 48 × *value of pd_module_trace_max operand* (bytes)

In the 64-bit mode: 64 + 64 × *value of pd_module_trace_max operand* (bytes)

23) pd_module_trace_timer_level = 0 | 10 | 20

Specifies how to acquire the time to be output in module traces. The following table explains the meaning of the value specified for this operand.

Specified value	Time acquisition method
`0`	Time is output in seconds at every module trace output location.
`10`	Time is output in microseconds only at performance-critical module trace output locations, such as those before and after input/output processing, and time is output in seconds at other locations.

Specified value	Time acquisition method
`20`	Time is output in microseconds at every module trace output location.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

If this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. If the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `0`.

Note

If you specify a value other than `0` for this operand, a function for acquiring time in microseconds is issued, and as a result, system performance might decline.

24) pd_pth_trace_max = *maximum-number-of-stored-communication-traces*

~<unsigned integer>((1024-8388608))

Specifies the maximum number of communication trace records to be used as troubleshooting information.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reasons such as performance checking, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `1024`.

Notes

Increasing the value of this operand increases the amount of process private memory secured by HiRDB processes.

Process private memory for communication traces is calculated based on this operand's value rounded up to the power of two. For details about memory requirements, see *Calculation of required memory* in the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_pth_trace_max` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

7.2.9 Operands related to global buffers

25) pd_max_add_dbbuff_no = *maximum-global-buffers-count-for-dynamic-addition*

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, this operand specifies the maximum number of global buffers (per server) that can be added dynamically by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

- Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify a sufficient value based on that value.
- Determine the operand's value in such a manner that the following condition is satisfied:
  Value of pd_max_add_dbbuff_no ≤ 2,000,000 - *number of global buffers allocated per server during HiRDB startup*

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the following value is assumed:

Condition		Default value
32-bit mode	$a \geq 500$	256
	$a < 500$	500 - *a*
64-bit mode	$a \geq 1{,}000$	256
	$a < 1{,}000$	1,000 - *a*

a: Number of global buffers allocated per server during HiRDB startup

Notes

Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the value of S* under *Determining the size of status files*
- *Formula 2*, *Formula 3*, *Formula 4*, and *Formula 5* under *Formulas for the size of the shared memory used by a dictionary server*

26) pd_max_add_dbbuff_shm_no = *maximum-shared-memory-segments-count-for-dynamic-addition*

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, specifies the maximum number of shared memory segments (per server) that can be allocated when dynamic addition is performed by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify an appropriate value.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, the following value is assumed:

Condition		Default value
`pd_max_add_dbbuff_no` operand is omitted.	32-bit mode	500 + *A*
	64-bit mode	1,000 + *A*
`pd_max_add_dbbuff_no` operand is specified.	32-bit mode	$\downarrow$ *value of pd_max_add_dbbuff_no* $\times$ 1.5 + *A* $\downarrow$ (If the value is 32,752 or greater, `32752` is set)
	64-bit mode	

A: Remaining number of shared memory segments that can be allocated during HiRDB startup. This value can be obtained from the following formula:

$A = a - b$

Assign the following values to *a* and *b*:

a = value of pd_max_dbbuff_shm_no (in 64-bit mode, `16`)

b = number of shared memory segments allocated to each server during HiRDB startup

You can obtain information about the shared memory segments by using the `pdls -d mem` command or an OS command.

Notes

- If the following condition is satisfied, the value of the `pd_max_add_dbbuff_no` operand is assumed in this operand:

  Value of `pd_max_add_dbbuff_shm_no` < value of `pd_max_add_dbbuff_no`

  The value of the `pd_max_add_dbbuff_no` operand is also assumed when the default value satisfies the above condition.
- Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.
- If the size of a shared memory segment to be added exceeds the `SHMMAX` operand value, shared memory is divided into multiple segments based on the `SHMMAX` operand value as the maximum value. Either increase the value of the `SHMMAX` operand based on the size of the shared memory segment to be added or increase the value of the `pd_max_add_dbbuff_shm_no` operand so that no shortage occurs when the shared memory is segmented.
- If the facility for dynamically changing global buffers is used, the total number of shared memory segments that are allocated for the global buffers can be obtained from the following formula:

  pd_max_dbbuff_shm_no value + pd_max_add_dbbuff_shm_no value

  Therefore, if more shared memory segments were allocated when HiRDB started than the number specified for `pd_max_dbbuff_shm_no`, the number of shared memory segments that are actually allocated during dynamic change is the value of `pd_max_add_dbbuff_shm_no` minus the excess number of shared memory segments.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_no`
- `pd_max_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_shm_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

7.2.10 Operands related to Java

Java operands are specified when a Java stored procedure or Java stored function is used. For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide.*

27) pd_java_stdout_file = *"Java-virtual-machine-standard-output-and-standard-error-output-destination-file"*
~<path name>

Specifies as an absolute path name the file to which the standard output and standard error output are to be output in a Java virtual machine.

Specification guideline

Because the size of the file specified by this operand is extremely large, this operand is not normally specified. It is recommended that this operand be specified during debugging of a Java stored procedure or Java stored function. There is no limit to the size of the file that can be specified by this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the Java Virtual Machine standard output and standard error output are ignored.

Note

If there are simultaneous writing attempts from multiple processes, their output contents cannot be guaranteed.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

28) pd_java_castoff = Y | N

Specifies whether to use the following events as triggers for shutting down the process at the server (single server, front-end server, dictionary server, or back-end server) that started the Java Virtual Machine:

No.	Server type	Process name	Trigger that ends process
1	Single server	`pdsds`	UAP is disconnected
2	Front-end server	`pdfes`	UAP is disconnected
3	Dictionary server	`pddic`	Transaction is completed or UAP is disconnected
4	Back-end server	`pdbes`	Transaction is completed or UAP is disconnected

`Y`: Shut down server process when trigger occurs.

`N`: Do not shut down server process when trigger occurs.

Specification guidelines

Normally, this operand is not specified. However, if you encounter the problems described below, we recommend that you specify `Y` in this operand:

- Use of the Java Virtual Machine causes the amount of memory usage to increase to the point where the available system memory becomes nearly exhausted.
- SQL code that includes numerous search conditions is executed, and even though the connection does not use the Java Virtual Machine, the maximum stack size set by the Java Virtual Machine on another connection prevents the stack from expanding, causing the server process to be aborted by a segmentation error.

For details about the Java Virtual Machine facility, see the Java Virtual Machine documentation.

Notes

On systems that run Java stored routines frequently, specifying `Y` in this operand will generate overhead for server process restarts and Java Virtual Machine startups.

Relationship to other operands

This operand is related to the `pd_process_count` operand.

7.2.11 Operands related to system log files

29) pd_log_dual = Y | N

Specifies whether dual system log files are to be used.

`Y`: Use dual system log files.

`N`: Do not use dual system log files.

Advantages

When dual system log files are used, HiRDB collects the same system log information in both files, which are called File A and File B. If a failure occurs in one of the files while the collected system log file is being loaded, the file can be loaded from the other file, resulting in higher system reliability.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

When use of dual system log files is specified, the name of the File B system log file must be specified with the `pdlogadpf` operand.

30) pd_log_remain_space_check = warn | safe

Specifies the processing to be performed by HiRDB when the available space in the system log file falls below the warning level. This is called the facility for monitoring the free area for the system log file. For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

`warn`:

When the available space in the system log file falls below the warning level, the `KFPS01162-W` message is output.

`safe`:

When the available space in the system log file falls below the warning level, scheduling of new transactions is suppressed, all transactions inside the server are forcibly terminated, and the `KFPS01160-E` message is output.

Specification guideline

We recommend that you specify `safe` because it can reduce the probability of abnormal termination of units due to system log file space shortage. However, when `safe` is specified, all transactions inside the server are forcibly terminated when a shortage occurs in the available space in the system log file. Therefore, the design of the system log file requires more accuracy. For details about system log file design, see the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `warn`.

31) pd_log_auto_unload_path = "*unload-log-file-output-directory*"[,"*unload-log-file-output-directory*"]...
~<path name>((1-136 characters))

Specifies as absolute path names the unload file output directories when the automatic log unloading facility is to be used for the system log. A HiRDB file system area name must be specified to create the unload log file in a HiRDB file system area. *The directories or HiRDB file system areas specified for this operand must be created before HiRDB is started.*

Additionally, in this operand, specify a different directory or HiRDB file system area for each server.

For details about the automatic log unloading facility, see the *HiRDB Version 9 System Operation Guide*.

Specification guidelines

It is important to check the available disk space before specifying a directory, so as to ensure that the created unload log file does not cause a disk space shortage.

If an unload log file cannot be created in the specified directory because of a disk space shortage, the automatic log unloading facility stops. If this is a possibility, creation of multiple directories is recommended.

Note, however, that the database recovery operation of selecting the unload files needed for recovery is simplified somewhat when only one directory is used.

Also keep in mind the following when multiple directories are created:

- It is recommended that directories be specified in different partitions to protect against disk errors.
- If the unload log file cannot be created in a single directory because of a full disk or disk error, create an unload log file under a different directory. HiRDB uses the directories specified by this operand in the order of their specification.

Operand rules

- Up to 128 directories can be specified.

- When multiple directories are specified, the same path name cannot be specified.

Notes

- The automatic log unloading facility cannot be used in the following cases:
  - `N` is specified in the `pd_log_unload_check` operand.
- If two or more directories are created for unload log files and there is no empty directory when HiRDB starts normally, the automatic log unloading facility stops.
- When a multi-HiRDB is being used, you must create a different directory for each HiRDB. Specifying the same directory for more than one HiRDB will make it impossible to determine which unload log file applies to which HiRDB.

32) pd_log_auto_unload_restart = Y | N

Specifies whether the automatic log unloading facility is to be restarted when the system log files are swapped following the issuance of a message indicating that the automatic log unloading facility stopped due to an error (message `KFPS01150-E`).

`Y`: Restart the automatic log unloading facility.

`N`: Do not restart the automatic log unloading facility.

Condition

The following two conditions must be satisfied:

- `Y` is specified in the `pd_log_unload_check` operand or this operand is omitted.
- The `pd_log_auto_unload_path` operand is specified.

Advantages

- If `Y` is specified in this operand and a shortage of disk capacity occurs due to an increase the number of work files, or because the unload processing fails due to a temporary error such as a process creation error, HiRDB automatically restarts the unloading of system logs the next time the system log files are swapped.
- If `N` is specified in this operand and the unload log files are set to be saved while the automatic log unloading facility is stopped due to an error, you can prevent the unload log files from being saved before the `pdlogatul -t` command is executed.

Specification guidelines

Normally, specify `Y` or omit this operand.

Specify `N` if you have already set HiRDB to monitor the automatic log unloading facility termination message (`KFPS01150-E` message) and restart the facility with the `pdlogatul -b` command.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `Y` is assumed.

33) pd_log_singleoperation = Y | N

This operand is applicable when dual system log files are used; it need not be specified when dual system log files are not used.

Specifies whether the single-operation mode is to be used for the system log files. Even if an error occurs in a system log file, making no dual system log files available, HiRDB (or a unit for a HiRDB/Parallel Server) can continue processing using the remaining single normal system log file without being abnormally terminated. This is called *single operation of the system log files*.

The mode in which continuation of processing is permitted only with both system log files available (normal operation mode) is called *double operation of the system log files*.

`Y`: Use single operation of the system log files.

`N`: Do not use single operation of the system log files. Both system log files must always be used.

Condition

This operand is valid only when `pd_log_dual = Y` is specified.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

34) pd_log_rerun_reserved_file_open = Y | N

Specifies whether a system log file is to be opened automatically.

If no system log file that can be overwritten is available during a unit restart, HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of system log file*.

A reserved file is used in the following cases:

- Between the time of a restart and the time when the first synchronization point dump is collected
- When none of the opened file groups can be overwritten

`Y`: Open a system log file automatically (open and use a reserved file).

`N`: Do not open a system log file automatically (do not use a reserved file).

Advantages

When `Y` is specified, the unit can be restarted as long as a reserved file is available, even though no file that can be swapped in is available during the unit restart.

However, if a file that is in unload wait status is available, the unit is stopped; once that file has been unloaded, the unit can be restarted.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

35) pd_log_rerun_swap = Y | N

Specifies whether the system log files are to be swapped during a unit restart.

`Y`: Swap the system log files.

`N`: Do not swap the system log files.

Advantage

When `Y` is specified, there can be a physical separation between the system log files used before and after a restart. Therefore, the system log file that was being used before the restart can be reused during server operation.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

36) pd_log_swap_timeout = *wait-time-for-completion-of-system-log-file-swapping*

~<unsigned integer>((1-32580)) (seconds)

Specifies in seconds the wait time during which swapping of system log files must be completed. If a swap of system log files is not completed within the specified amount of time, the unit terminates abnormally.

Specification guidelines

Normally, there is no need to specify this operand. If it takes a long time to swap system log files for a reason such as poor machine performance, you can specify this operand with a value that is greater than the default value. To detect errors or delays quickly during system log file swapping so that the unit can be terminated (such as because of a disk failure), reduce the value of this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `180`.

37) pd_log_unload_check = Y | N

Specifies whether HiRDB is to check the unload status of system log files.

`Y`:

Check the unload status (normal operation).

`N`:

Do not check the unload status. A system log file is placed in swappable status, regardless of its unload status, when both of the following conditions are satisfied:

- It is in overwritable status
- It is in extraction completed status (HiRDB Datareplicator)

In such a case, the system log file operation method is to release checking of unload status. For details about this operation method, see the *HiRDB Version 9 System Operation Guide*.

Advantages

Specifying `N` provides the following advantages:

- Operations are simplified because the system log file unload operation is eliminated.
- It is not necessary to provide files for storing unload files.

Specification guideline

Specify `N` if the system log file will not be needed for database recovery (in other words, if recovery from a backup collection point will be sufficient).

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `Y`.

Notes

The following points apply when `N` is specified:

- Database can be recovered only if backups have been made.
- If this option is specified when the system log file is required for database recovery, it will not be possible to recover the database.

38) pd_log_max_data_size = *log-input/output-buffer-size*

~<unsigned integer>((32000-523000)) (bytes)

Specifies in bytes the size of the buffer to be used for system log input/output operations.

Specification guideline

Specify a value that satisfies conditional expression 1 below. If `uap` is specified in the `pd_rpl_reflect_mode` operand or a recovery-unnecessary front-end server is used, specify a value that satisfies both the conditional expressions below (1 and 2). To optimize the value, use the tuning method for the specification value.

Conditional expression 1: log input/output buffer length $\geq$ *a*

a: 72 $\times$ (*maximum number of back-end and dictionary servers that are targets for reference or update processing by a single transaction*) + 1,344

Conditional expression 2: log input/output buffer length $\geq$ *b*

b: (*maximum number of back-end and dictionary servers that are targets for reference or update processing by a single transaction* + 1) $\times$ 128 + 64

Tuning the specified value

A value (other than the default value) might need to be specified for this operand after the following types of statistics analysis utility statistical information related to system operation have been checked:

- Number of buffer sectors waiting for input/output (`# OF BUFFER FOR WAIT I/O`)

  If the average number of buffer sectors waiting for input/output significantly exceeds 100, increase the value for this operand so that the average approaches 100.
- Number of waits caused by lack of a current buffer (`# OF WAIT THREAD`)

  If the number of waits caused by lack of a current buffer is not 0, increase the value for this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `400000` (however, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `32000`).

Notes

The specification of this operand affects the response and throughput of SQL code executed by a transaction. When a small value is specified, writing to system log files occurs frequently, which might result in deterioration of performance.

Relationship to other operands

Use this operand and the `pd_log_write_buff_count` operand to determine the log I/O buffer size.

39) pd_log_write_buff_count = *log-output-buffer-sectors-count*

~<unsigned integer>((3-65000))

Specifies the number of buffer sectors to be used for system log output.

Tuning the specified value

Specify `10` (the default) for this operand initially. Thereafter, use a statistics analysis utility to obtain statistical information related to system operation to check the number of waits caused by a shortage of current buffer space (`# OF WAIT THREAD`). If the number of waits is high, specify a larger value in order to improve throughput.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `10` (however, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `3`).

Notes

If a small value is specified in this operand and the number of transactions is high, multiple transactions might be waiting for system log output, lowering system performance.

Relationship to other operands

Use this operand and the `pd_log_max_data_size` operand to determine the log output buffer sector count.

40) pd_log_rec_leng = *system-log-file-record-length*

~<unsigned integer>((1024, 2048, 4096)) (bytes)

Specifies the record length for the system log files; the specifiable values are `1024`, `2048`, and `4096`.

Specify the record length specified in the `-l` option of the `pdloginit` command for this operand.

Specification guidelines

Specify the record length based on the guidelines for designing the record length of system log files. For details about the guidelines for designing the record length of system log files, see *Record length of a system log file* in the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `4096`.

Notes

- If a value that is different from the record length specified by the `-l` option of the `pdloginit` command is specified for this operand, system log files cannot be opened.
- For details about how to modify the system log file record length, see the *HiRDB Version 9 System Operation Guide*.

41) pd_log_rollback_buff_count = *rollback-log-input-buffer-sector-count*

~<unsigned integer>((0-256))

Specifies the number of buffer sectors to be used for system log input during rollback processing. When `0` is specified in this operand, HiRDB determines the number of rollback log input buffer sectors.

Specification guidelines

- we recommend that you specify `0` in this operand, in which case HiRDB will calculate an appropriate value automatically.
- If the specified value is too small, concurrent execution of rollbacks might be slowed. If the specified value is too large, the unit controller might use more shared memory than is necessary.

Tuning the specified value

Specify `0` in this operand. If specifying `0` leads to memory shortages, do not specify this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is as shown below:

Number of servers in unit $\times$ 2

Effects on individual estimation formulas

If the value of the `pd_log_rollback_buff_count` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

42) pd_log_auto_expand_size = ***extension-amount-per-system-log-file-extension-trigger*****[,*****extension-limit*****]**

~<unsigned integer>((0-104857600))<<0,0>>(records)

Specify this operand when the system log file automatic extension facility is used.

Specifies the number of records to be added to the system log file for each extension trigger and an upper limit for extending file size.

If the amount to be added per trigger is omitted or `0` is specified, the system log file will not be extended automatically. If the extension limit is omitted or `0` is specified, the system log file might be extended until either the disk on which the file system area is located becomes full or the system log file capacity reaches its upper limit. When a value is specified for the amount to be added per trigger that is larger than the extension limit, the file will be extended up to the extension limit.

For details about the system log file automatic extension facility, see the *HiRDB Version 9 System Operation Guide*.

Condition

`Y` must be specified in the `pd_large_file_use` operand or specification of this operand must be omitted.

Specification guidelines

- Specify in this operand the amount to be added per trigger based on the number of records specified when the system log file was created with the `-n` option of the `pdloginit` command. Determine 10 percent of the average number of records for all system log files and specify that value.
- Normally, an extension limit is omitted.

Tuning the specified value

If the system log output volume exceeds the expanded size after automatic extension, the system log file could become full, resulting in a unitdown. In such a case, increase the specified value (number of records to be added per trigger). If extension processing requires so much time that it affects transaction performance, specify a smaller value.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed.

7.2.12 Operands related to synchronization point dump files

43) pd_spd_dual = Y | N

Specifies whether to use dual synchronization point dump files.

`Y`: Uses dual synchronization point dump files.

`N`: Does not use dual synchronization point dump files.

Advantage

When dual synchronization point dump files are used, HiRDB collects the same synchronization point dump in both dump files A and B. Even when an error occurs in one of the files when the collected synchronization point dump is being loaded, the synchronization point dump can be loaded from the other file, thus improving system reliability.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `N` is assumed.

Relationship to other operands

To use dual synchronization point dump files, specify the name of synchronization point dump file B in the `pdlogadpf` operand.

44) pd_spd_assurance_msg = Y | N

Specifies whether the KFPS02183-I message is to be output when a synchronization point dump is completed.

Y: Output the message.

N: Do not output the message.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, Y is assumed.

45) pd_spd_assurance_count = *number-of-guaranteed-valid-generations*

~<unsigned integer>((1-2))

Specifies the range of system log files to be saved during HiRDB operation as a protection against events such as a synchronization point dump file input error during system recovery. What is specified here is a number of synchronization point dump file generations; the specified value is referred to as the *number of guaranteed-valid generations*. The number of past generations of synchronization point dump files specified here are overwrite disabled.

Advantage

When 2 is specified as the number of guaranteed-valid generations and an error occurs in the most recent synchronization point dump file generation, the system can be recovered using the preceding synchronization point dump file generation, resulting in higher reliability.

Specification guidelines

- To improve reliability, specify 2 for the number of guaranteed valid generations. However, the number of overwrite disabled synchronization point dump files increases (to two).
- If reliability has been improved with the use of dual synchronization point dump files, we recommend that you omit this operand or specify 1 for it.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, 1 is assumed.

Notes

- The minimum number of synchronization point dump files needed is *number-of-guaranteed-valid-generations* + 1.
- Specifying 2 increases the number of overwrite disabled synchronization point dump files. Moreover, the system log files that correspond to the overwrite disabled synchronization point dump files are also overwrite disabled. Consequently, specifying 2 for the number of guaranteed valid generations increases the number of overwrite disabled system log files. As a result, a shortage might occur in the number of system log files that can be swapped in. To prevent this, it might be necessary to re-evaluate the system log file capacity.

46) pd_spd_reduced_mode = *reduced-mode-operation-option*

~<unsigned integer>((0-2))

Specifies whether the reduced mode operation for synchronization point dump files is to be used.

Reduced mode operation is a facility for continuing processing if at least two synchronization point dump files can be used, even if an event such as a file error during HiRDB operation or restart reduces the number of synchronization point dump files to or below the number of required files (*number of guaranteed valid generations*# + 1).

#: Value specified for the pd_spd_assurance_count operand.

0: Do not use the reduced mode operation.

1: Use the reduced mode operation.

2: Use the reduced mode operation and issue a warning message whenever a synchronization point dump is being acquired during the reduced mode operation.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, 0 is assumed.

47) pd_spd_reserved_file_auto_open = Y | N

Specifies whether a synchronization point dump file is to be opened automatically. When `Y` is specified and an error in a synchronization point dump file reduces the number of synchronization point dump files to the number of files necessary for operation (*number of guaranteed valid generations*# + 1), HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of synchronization point dump file*.

#: Value specified for the `pd_spd_assurance_count` operand.

`Y`:

Open a synchronization point dump file automatically. A reserved file is opened when the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

`N`:

Do not open a synchronization point dump file. No reserved file is opened, even though the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `N` is assumed.

Relationship to other operands

This operand has a higher priority than the `pd_spd_reduced_mode` operand.

48) pd_spd_max_data_size = *synchronization-point-dump-file-buffer-size*
~<unsigned integer>((32000-4000000)) (bytes)

Specifies in bytes the size of the buffer (shared memory) to be used for synchronization point dump file input/output operations.

The value specified here affects the number of synchronization point dump file input/output operations.

Specification guidelines

- Normally, this operand need not be specified.
- When the value of this operand is increased, the number of synchronization point dump file input/output operations is reduced.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `32768` is assumed.

49) pd_log_sdinterval = *system-log-output-volume*[,interval]

Specifies the collection interval for synchronization point dumps. This operand can be specified based on the following information:

- Volume of system log information
- Amount of time that has elapsed since the previous synchronization point dump was collected

***system-log-output-volume*: ~<unsigned integer>((100-100000)) (number of log blocks)**

Specifies an interval between synchronization point dumps in terms of the number of blocks of log information. A synchronization point dump is collected each time system log information equivalent to the number of log blocks specified here has been output.

***interval*: ~<unsigned integer>((0 or 10-1440)) (minutes)**

Specifies a synchronization point dump collection interval in terms of the number of minutes between synchronization point dumps. A synchronization point dump is collected when the time interval specified here has elapsed since the previous synchronization point dump was collected.

- If `0` is specified for the interval, HiRDB does not use a time interval for collecting synchronization point dumps.
- If no transactions execute during the time interval since the previous synchronization point dump was collected, no synchronization point dump is collected even though the amount of the time specified here has elapsed.

Specification guidelines

- This operand need not be specified if no specific amount of time is specified for restarting HiRDB.

- The value specified for this operand affects the amount of time required to restart HiRDB.

  Specifying a small value for this operand reduces the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps increases, online performance might deteriorate in some cases.

  Conversely, specifying a large value for this operand increases the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps decreases, online performance might improve in some cases.

Tuning the specified value

The synchronization point dump collection intervals can be checked with the statistics analysis utility; the relevant information is shown under `SYNC POINT GET INTERVAL` in the statistical information related to system operation. Use the average of the `SYNC POINT GET INTERVAL` values. If the synchronization point dump collection interval is determined to be too long, decrease the specification value; conversely, if it is determined to be too short, increase the specification value.

Operand default value

If this operand is omitted, the values specified for the same operand in the server common definition take effect. If the same operand is also omitted from the server common definition, the following values are assumed:

- *system-log-output-volume*: `5000`
- *interval*: `60`

Note if `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, `1000` is assumed as the default value for the system log output volume.

Note

- The synchronization point dump collection interval is determined on the basis of the volume of system log information that is output. Therefore, committing data from memory to a database takes a long time during intervals that have few updating transactions. If an error occurs during such a time, it will take longer to recover the transactions that were generated during that period. If this is a possibility, also use the `interval` value to set the synchronization point dump collection interval.
- When synchronization point dumps are collected, update pages are output from the global buffer, proportionately increasing the loads on the CPU, input/output processing, and lock time. For this reason, reducing the synchronization point dump collection interval might cause processing delays.

7.2.13 Operands related to server status files

50) pd_sts_file_name_1 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

:

pd_sts_file_name_7 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

Define server status files. Although the `pd_sts_file_name_2` to `7` operands can be omitted, the `pd_sts_file_name_1` operand cannot be omitted.

"*logical-file-name*" ~<identifier>((1-8 characters))

Specifies the logical file name of a status file for the dictionary server.

When a command that manipulates the status file is to be executed, the logical file name defined here must be specified.

"*file-a-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File A status file as an absolute path name.

"*file-b-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File B status file as an absolute path name.

Specification guidelines

- The files specified as File A and File B must be status files created with the `pdstsinit` command. If a file that has not been created with the `pdstsinit` command is specified, a virtual status file is created.
- If an error occurs in a status file, HiRDB swaps the status files. If no spare file is available for swapping, HiRDB (or a unit for a HiRDB/Parallel Server) terminates abnormally. For this reason, defining a large

number of system files improves system reliability, but at the expense of increasing the amount of disk space that is required.

- The status files specified for File A and File B must have the same record length and capacity.

Operand rules

- Up to seven instances of this operand can be specified.
- A status file has a dual structure consisting of File A and File B; both must be specified.
- Environment variables cannot be used for the absolute path names of the File A and File B file names.
- The same names cannot be specified for the logical file name, the File A file name, and the File B file name.
- HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sts01`, `C:\hirdb\sysfile` is not case sensitive, but `sts01` is case sensitive.

Notes

- When HiRDB is started normally, the primary file (the file that was the primary file when HiRDB was terminated) is inherited. However, if there is no current file that can be inherited, for example because all status files have been initialized, the first status file that was specified among those specified by the `pd_sts_file_name_1` to `7` operands becomes the current file, and the remaining files become spare files if they can be opened. Any files that cannot be opened become reserved files.
- When HiRDB is restarted, the primary file (the file that was the primary file when HiRDB was terminated) is inherited.

Using a virtual status file

Specifying a virtual status file makes it possible to add a new status file while HiRDB is running.

For example, if the number of spare files is reduced because of errors in status files, virtual status files can be converted into spare files. The procedure for converting virtual status files into spare files follows.

Procedure

1. Use the `pdstsinit` command to create a status file in a HiRDB file system area for system files.
2. Use the `pdstsopen` command to open the status file.

These operations can be performed while HiRDB is running; there is no need to stop HiRDB.

- **Advantages and disadvantages**

  Defining a virtual status file reduces the amount of space required in the HiRDB file system area. However, a virtual status file cannot be added as a spare file if the HiRDB file system area for system files does not have sufficient space (sufficient space for adding the file), thus reducing system reliability.

  When virtual status files are not defined, the amount of space required in the HiRDB file system area is increased. However, because a swap destination is guaranteed when a file error occurs, system reliability is increased.

- **Notes**

  When virtual status files are defined, HiRDB determines that a status file error has occurred when HiRDB is started. For this reason, HiRDB cannot start if `stop` (default value) is specified for the `pd_sts_initial_error` operand. When virtual status files are defined, specify `continue` or `excontinue` for the `pd_sts_initial_error` operand. It is also necessary before starting HiRDB to specify the current file in the `pd_sts_last_active_file` operand.

7.2.14 Operands related to server status files (when an error occurs)

For details about the measures to be taken when an error occurs in a status file, see the *HiRDB Version 9 System Operation Guide*.

51) pd_sts_initial_error = stop | continue | excontinue

When a server (single server, front-end server, dictionary server, or back-end server) starts, HiRDB performs a process of identifying the current server status file. This operand specifies the action that HiRDB takes when any of the following errors are detected during this identification process.

- No real server status file is found.

- An error is detected in the server status file.

The current file identification process is applied to the server status files specified by the `pd_sts_file_name_1` to `7` operands.

`stop`:

When an error is detected in a server status file during the current file identification process, the startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started. In this case, first take a corrective action for the status file in which the error was detected, and then start HiRDB.

`continue` or `excontinue`:

Even when an error is detected in the server status file during the current file identification process, the startup of the server is continued if the current file is normal. However, the startup might be stopped depending on the value specified for the `pd_sts_singleoperation` operand (whether operation continues with a single status file). The following table shows the relationship to the `pd_sts_singleoperation` operand.

- **Relationship to the pd_sts_singleoperation operand**

pd_sts_singleoperation operand value	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
`continue`	When an error is detected in a server status file, HiRDB cannot identify the current file, and thus startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	The HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.
`stop` (default value)	When an error is detected in a server status file, HiRDB identifies the current file and startup of the server is continued. However, if the primary and secondary files satisfy any of the conditions listed in the table below (cases in which HiRDB cannot identify the current file), startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	If HiRDB cannot identify the current file, the HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.

- **When HiRDB cannot identify the current file**

pd_sts_initial_error operand value	File A status	File B status
`continue`	Error shutdown	Error shutdown
	Error shutdown	Open (initial state)
	Error shutdown	No real file
	Open (initial state)	Error shutdown
	Open (initial state)	No real file
	No real file	Open (initial state)
	No real file	Error shutdown
	No real file	No real file
`excontinue`	Error shutdown	Error shutdown
	Error shutdown	No real file
	No real file	Error shutdown
	No real file	No real file

Therefore, to minimize actions by the HiRDB administrator (to reduce the number of cases in which both the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands must be specified), specify the following:

- `pd_sts_initial_error = excontinue`
- `pd_sts_singleoperation = stop`

Specification guidelines

The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_initial_error operand value	
	stop	continue or excontinue
Processing by HiRDB during server startup	When an error is detected in a server status file, startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	Even when an error is detected in a server status file, the startup of the server is continued if the current file is normal.
Specification guideline	To improve system reliability, specify `stop`.	To simplify the error-handling actions during HiRDB startup, specify `continue` or `excontinue`.
Advantage	Guarantees that all server status files of the server are normal when the server starts. Therefore, if an error occurs in the current file after HiRDB has started, it can be swapped to a spare file.	Even when an error is detected in a server status file during server startup, HiRDB can start with the remaining normal files only. Therefore, HiRDB stop time can be shortened. In this case, because the number of spare files has become small, it is necessary to immediately repair the status files containing errors.
Disadvantage	Possibility increases that an error in a server status file stops the startup of HiRDB.	Because HiRDB might be running with only a small number of spare files, system reliability is low. Depending on the number of spare files available, it might not be possible to swap server status files.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `stop` is assumed.

Notes

- If both current files are abnormal, startup of the server is stopped regardless of the value specified for this operand, and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.
- Before starting HiRDB, do not initialize the current file with the `pdstsinit` command. If the current file is initialized, HiRDB cannot be restarted.
- If `excontinue` is specified for this operand and all generations of the status files are in single-operation mode and open (initial status), HiRDB does not start.

Remarks

The following figure shows the values specified for this operand, the processing performed by HiRDB, and the actions to be taken by the HiRDB administrator.

- Values specified for this operand and the processing performed by HiRDB

pd_sts_initial_error operand value	Status file error	pd_sts_singleoperation operand value	Can HiRDB identify the current file?	Specification of pd_sts_last_active_file	Does the pd_sts_last_active_file operand value match the latest file that can be opened?	Current file error	Specification of the pd_sts_last_active_side operand	Is the file specified for the pd_sts_last_active_side operand usable?	Numbers listed in the Table Processing performed by HiRDB and actions to be taken by HiRDB the administrator
`stop` (default value)	None	—	—	—	—	—	—	—	[1]
	Yes	—	—	—	—	—	—	—	[5]
`continue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
`excontinue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]

Legend:

—: Not applicable (the condition does not affect the processing performed by HiRDB)

- Processing performed by HiRDB and the actions to be taken by the HiRDB administrator

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[1]	HiRDB startup processing continues.	None
[2]	HiRDB identifies the latest current file and continues the startup processing.	Make the file that is in the error-shutdown state into a spare file.
[3]	Using the file specified in the `pd_sts_last_active_file` operand as the current status file, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[4]	Using the files specified in the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands as the current status files, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[5]	Because `stop` is specified for the `pd_sts_initial_error` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000010` in message `KFPS01005-E`.

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[6]	Because the current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000016` in message `KFPS01005-E`.
[7]	Because the normal current file identified by HiRDB does not match the file specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000017` in message `KFPS01005-E`.
[8]	Because the current file name identified by HiRDB does not match the file name specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000015` in message `KFPS01005-E`.
[9]	Because the normal current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000018` in message `KFPS01005-E`.

52) pd_sts_singleoperation = stop | continue

Specifies whether processing of server status files continues in the single-operation mode.

The status file single-operation mode means that processing is to continue using only the normal file (single file) when an error occurs in the status file and a spare file is not available. For details about the status file single-operation mode, see the *HiRDB Version 9 Installation and Design Guide*.

If an error occurs in one of the current files and a spare file is available, the status file is swapped and processing continues regardless of the value specified for this operand (operation in the single-operation mode does not occur).

`stop`:

Do not permit operation in the single-operation mode. If operation in the single-operation mode is necessary, HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated. If HiRDB is abnormally terminated, allocate spare files and then restart HiRDB.

`continue`:

Enable operation in the single-operation mode. When the single-operation mode goes into effect, the message `KFPS01044-I` is output. If an error occurs in the normal file during operation in the single-operation mode or if HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated while the status file is being updated, HiRDB cannot be restarted. Therefore, when the single-operation mode goes into effect, immediately allocate spare files.

Specification guidelines

- We recommend that you specify `stop` to increase system reliability. We also recommend that you increase the number of spare files to guard against errors in the current files.
- The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_singleoperation operand value	
	stop	continue
Specification guideline	To improve system reliability, specify `stop`.	Specify `continue` if it is important not to stop HiRDB.
Advantage	When an error occurs in one of the current files and a spare file is not available, operation in the single-operation mode does not occur and HiRDB is abnormally terminated. Consequently, the possibility of losing the content of the current files is reduced.	Even when an error occurs in one of the current files and a spare file is not available, processing can be continued. Therefore, the possibility that an error in the status file stops HiRDB is reduced.
Disadvantage	Possibility that an error in the status file stops HiRDB increases. However, increasing the number of spare files can reduce this possibility.	If an error occurs in the normal status file during operation in the single-operation mode or if HiRDB is abnormally terminated during updating of the status file, the content of the current file is lost and HiRDB cannot be restarted.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `stop` is assumed.

Relationship to other operands

The combination of the values specified for the `pd_sts_singleoperation` and `pd_sts_initial_error` operands determines the processing by HiRDB when an error occurs in the status file. Therefore, the values to be specified for these operands must be determined together.

53) pd_sts_last_active_file = "*logical-file-name*"

~<identifier> ((1-8 characters))

Specifies the name of the logical file to be used as the current status file at the time of the front-end server startup. HiRDB compares the file name specified in this operand with the file name selected by HiRDB to be the current file. If the file names match, HiRDB (or a unit for a HiRDB/Parallel Server) is started; otherwise, HiRDB (or a unit for a HiRDB/Parallel Server) is not started.

Conditions

The following conditions must be satisfied:

- `continue` or `excontinue` is specified in the `pd_sts_initial_error` operand.
- It cannot be determined if the current file selected by the HiRDB system was the most recent current file during the previous session.

Specification guidelines

1. To start HiRDB immediately after initializing all status files:

   From among the operable logical files specified in the `pd_sts_file_name1` to `7` operands, specify the one that has the smallest number. Forced startup will be used in this case, regardless of the previous termination mode.

2. When both of the current files are normal:

   Specify the name of the current file.# If HiRDB cannot be started even though the name of the current file is specified, the current file might have been initialized. In this case, first initialize all status files, and then use the method in step 1 above to start HiRDB (forced startup will be used, regardless of the previous termination mode).

3. When one of the current status files has an error:

   Use the method in step 2 above, with the following operands specified:

   - `pd_syssts_singleoperation = continue`
   - `pd_sts_last_active_side`

4. When both of the current status files have errors:

   Initialize all status files, then execute the method in step 1 above (forced startup will be used, regardless of the previous termination mode).

5. When a virtual status file is specified:

   Specify the name of the current file.#

#: The names of the current files (that were active at the end of the previous operation) can be determined from the following messages:

- `KFPS01001-I`
- `KFPS01010-E`
- `KFPS01011-I`
- `KFPS01063-I`

Of the status files displayed by these messages, the one that is reported in the message that was output most recently is the current file.

54) pd_sts_last_active_side = A | B

Specify this operand if you want to start the dictionary server when one of the current files is in an error state. Specify the normal status file for this operand. HiRDB compares the file specified in this operand with the file selected by HiRDB. If they match, HiRDB copies the contents of the normal status file to secondary File A and File B, then HiRDB switches the spare file in as the current file and starts the unit. If the files do not match, HiRDB (or a unit for a HiRDB/Parallel Server) is not started.

Conditions

The following operands must be specified:

- `pd_sts_initial_error = continue` or `excontinue`
- `pd_sts_last_active_file`

7.2.15 Operands related to work table files

55) pdwork -v "*HiRDB-file-system-area-name*"[,"*HiRDB-file-system-area-name*"]...

~<path name of up to 141 characters>

Specifies the names of HiRDB file system areas for work table files. Work table files are used for temporary storage of information during execution of SQL statements; they are created automatically by HiRDB. For details about the SQL statements that require work table files, see *Overview of the work table file* in the *HiRDB Version 9 Installation and Design Guide*.

You must not omit this operand, because doing so might prevent execution of SQL code that requires work table files.

Notes

- Specify in this operand the HiRDB file system area that was initialized using the `pdfmkfs` command.
- If the size of the work table file is large, specify a large HiRDB file system area. For details about how to estimate the work table file size, see the *HiRDB Version 9 Installation and Design Guide*.

  When a HiRDB file system area is set up initially with the `pdfmkfs -a` command, HiRDB will extend the area automatically whenever the amount of space specified in the `-n` option is used up. For details about the `pdfmkfs` command, see the manual *HiRDB Version 9 Command Reference*.
- The HiRDB file system areas for work table files must be different from the HiRDB file system areas for system files and RDAREAs.
- If more than one HiRDB file system area is specified in this operand, and one of these HiRDB file system areas generates an error when an attempt is made to create a work table file in it, that file system area will usually not be used thereafter. Instead, only the other specified HiRDB file system areas will be used.

  However, if subsequent attempts to create work table files fail in all of the other specified HiRDB file system areas (due to reasons such as insufficient space or an excessive numbers of files), creation of a work table file will be attempted in the HiRDB file system area that had been taken out of use. If a work table file can be created in it normally, that HiRDB file system area will then be used.

  Note that when a dictionary server is shut down and restarted, a HiRDB file system area that was not being used because it had generated an error during work table file creation becomes usable again.

Operand rules

- At least one HiRDB file system area must be specified.
- A maximum of 16 HiRDB file system areas can be specified.
- This operand can be specified only once in the dictionary server definition. If it is specified more than once, the first specification is effective.
- Do not specify a HiRDB file system area that is being used by a back-end server.

7.2.16 Operands related to system log file configuration

56) pdlogadfg -d sys -g *file-group-name* [ONL]

Specifies a file group for a system log file. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign system log files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies a name for the file group. All file group names must be unique within a server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 200 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 200 times.

Notes

Before adding, modifying, or deleting this operand when a HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while the HiRDB Datareplicator is running, its extraction process might fail.

57) pdlogadpf -d sys -g *file-group-name* -a "*system-log-file-name*" [-b "*system-log-file-name*"]

Specifies the system log files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file that comprises the file group. Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

-b "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file B when dual system log files are to be used (`pd_log_dual = Y` specified). If `pd_log_dual = Y` is not specified, the system log file name is invalid, even if it is specified.

Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\log01`, `C:\hirdb\sysfile` is not case sensitive, but `log01` is case sensitive.

Notes

Before adding, modifying, or deleting this operand when HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while HiRDB Datareplicator is running, its extraction process might fail.

7.2.17 Operands related to synchronization point dump file configuration

58) pdlogadfg -d spd -g *file-group-name* [ONL]

Specifies a file group for synchronization point dump files. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign synchronization point dump files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<<identifier>((1-8 characters))

Specifies a name for the file group. All file group names within a unit must be server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 30 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 60 times.

59) pdlogadpf -d spd -g *file-group-name* -a "*synchronization-point-dump-file-name*" [-b "*synchronization-point-dump-file-name*"]

Specifies the synchronization point dump files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name*: ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*synchronization-point-dump-file-name*": ~<path name> ((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file that comprises the file group. Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

-b "*synchronization-point-dump-file-name*": ~<path name> ((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file B when dual synchronization point dump files are to be used (`pd_spd_dual = Y` specified). If `pd_spd_dual = Y` is not specified, the synchronization point dump file name is invalid, even if it is specified.

Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sync01`, `C:\hirdb\sysfile` is not case sensitive, but `sync01` is case sensitive.

7.2.18 Operands related to Plug-ins

60) pdplgprm -n *plug-in-name* [-s *shared-memory-size*]

Specifies the name of a plug-in and the size of the memory to be shared by the plug-in. Omit this operand if no plug-ins are to be used or no plug-ins will run in dictionary server.

Conditions

The plug-in specified here must have been registered in HiRDB with the `pdplgrgst` command.

-n *plug-in-name*: ~<identifier>((1-30 characters))

For details about the names of plug-ins that can be specified here, see the manuals for the plug-ins.

-s *shared-memory-size*: ~<unsigned integer> ((1-2000000)) <<0>> (kilobytes)

Specifies in kilobytes the size of the shared memory to be used by the plug-in. For details about the size of the shared memory to be used by the plug-in, see the manual for the applicable plug-in.

Effects on individual estimation formulas

If the value of the `pdplgprm` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for the size of the shared memory used by a dictionary server*

8 Back-End Server Definition

This chapter explains the operands of the back-end server definition.

8.1 Operand formats

A back-end server definition defines information for a back-end server. This section explains the formats used to specify the operands of a back-end server definition. Note that the numbers in the following table correspond to the numbers assigned to the explanations of the individual operands in *8.2 Operand explanations*.

For users who are creating HiRDB system definitions for the first time

The first step is to determine the values to be specified for the operands shown in boldface type. In principle, HiRDB can be started once the boldface operands have been specified.

No.	Format	Operand category
1	[set pd_max_bes_process = *maximum-number-of-activated-processes-per-back-end-server*]#	Processes
2	[set pd_process_count = *resident-processes-count* [,*resident-processes-count-at-server-startup*]]#	
3	[set pd_server_cleanup_interval = *interval-for-stopping-nonresident-server-processes*]#	
4	[set pd_max_ard_process = *asynchronous-READ-process-count*]#	
5	[set pd_dfw_awt_process = *number-of-parallel-write-processes-for-deferred-write-processing*]	
6	[set pd_work_buff_mode = each | pool]#	Work tables
7	[set pd_work_buff_size = *work-table-buffer-size*]#	
8	[set pd_work_buff_expand_limit = *work-table-buffer-expansion-limit*]#	
9	[set pd_spd_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps*]#	System monitoring
10	[set pd_dfw_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*]#	
11	[set pd_lck_pool_size = *server-lock-pool-size*]#	Lock
12	[set pd_lck_pool_partition = *per-server-lock-pool-partition-count*]#	
13	[set pd_lck_until_disconnect_cnt = *total-number-of-tables-and-RDAREAs-to-be-locked-per-server-UNTIL-DISCONNECT-specification*]#	
14	[set pd_max_open_holdable_cursors = *maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*] #	
15	[set pd_lck_hash_entry = *lock-pool-hash-entry-count*]#	
16	[set pd_dbsync_lck_release_count = *global-buffer-lock-release-interval-during-synchronization-point-processing*]#	
17	[set pd_sql_object_cache_size = *SQL-object-buffer-size*]#	Buffers
18	[set pd_bes_shmpool_size = *back-end-server-shared-memory-size*]#	Shared memory
19	[set pd_rpc_trace = Y | N]#	RPC trace information
20	[set pd_rpc_trace_name = "*name-for-RPC-trace-collection-files*"]#	
21	[set pd_rpc_trace_size = *RPC-trace-collection-file-size*]#	

No.	Format	Operand category
22	`[set pd_module_trace_max` = *maximum-number-of-module-traces-that-can-be-stored*`]`#	Troubleshooting information
23	`[set pd_module_trace_timer_level` = 0 | 10 | 20`]` #	
24	`[set pd_pth_trace_max =` *maximum-number-of-stored-communication-traces*`]`#	
25	`[set pd_max_add_dbbuff_no =` *maximum-global-buffers-count-for-dynamic-addition*`]`#	Global buffers
26	`[set pd_max_add_dbbuff_shm_no =` *maximum-shared-memory-segments-count-for-dynamic-addition*`]`#	
27	`[set pd_max_temporary_object_no` = *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*`]`#	Temporary tables
28	`[set pd_plugin_ixmk_dir = "`*index-information-file-creation-directory-name*`"` *or* `"`*index-information-file-creation-HiRDB-file-system-area-name*`"]`#	Delayed batch creation of plug-in index
29	`[set pd_java_stdout_file = "`*Java-virtual-machine-standard-output-or-standard-error-output-destination-file*`"]`#	Java
30	`[set pd_java_castoff = Y | N]`#	
31	`[set pd_log_dual = Y | N]`#	System log files
32	`[set pd_log_remain_space_check = warn | safe]`#	
33	`[set pd_log_auto_unload_path = "`*unload-log-file-output-directory*`"` *[,*`"`*unload-log-file-output-directory*`"`]...]	
34	`[set pd_log_auto_unload_restart = Y | N]`#	
35	`[set pd_log_singleoperation = Y | N]`#	
36	`[set pd_log_rerun_reserved_file_open = Y | N]`#	
37	`[set pd_log_rerun_swap = Y | N]`#	
38	`[set pd_log_swap_timeout =` *wait-time-for-completion-of-system-log-file-swapping*`]`#	
39	`[set pd_log_unload_check = Y | N]`#	
40	`[set pd_log_max_data_size =` *log-input/output-buffer-size*`]`#	
41	`[set pd_log_write_buff_count =` *log-output-buffer-sectors-count*`]`#	
42	`[set pd_log_rec_leng =` *system-log-file−record-length*`]`#	
43	`[set pd_log_rollback_buff_count =` *rollback-log-input-buffer-sector-count*`]`#	
44	`[set pd_log_auto_expand_size =` *extension-amount-per-system-log-file-extension-trigger*`[,`*extension-limit*`]]`#	
45	`[set pd_spd_dual = Y | N]`#	Synchronization point dump files
46	`[set pd_spd_assurance_msg = Y | N]`#	
47	`[set pd_spd_assurance_count` = *number-of-guaranteed-valid-generations*`]`#	
48	`[set pd_spd_reduced_mode =` *reduced-mode-operation-option*`]`#	
49	`[set pd_spd_reserved_file_auto_open = Y | N]`#	
50	`[set pd_spd_max_data_size` = *synchronization-point-dump-file-buffer-size*`]`#	

No.	Format	Operand category
51	`[set pd_log_sdinterval =` *system-log-output-volume* `[,` *interval* `]]`#	
52	`set pd_sts_file_name_1 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"`	Server status files
	`[set pd_sts_file_name_2 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_sts_file_name_3 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_sts_file_name_4 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_sts_file_name_5 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_sts_file_name_6 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
	`[set pd_sts_file_name_7 = "`*logical-file-name*`" ,"`*file-a-status-file-name*`","`*file-b-status-file-name*`"]`	
53	`[set pd_sts_initial_error = stop | continue | excontinue]`#	Server status files (when an error occurs)
54	`[set pd_sts_singleoperation = stop | continue]`#	
55	`[set pd_sts_last_active_file = "`*logical-file-name*`"]`	
56	`[set pd_sts_last_active_side = A | B]`	
57	`[set pd_bes_connection_hold = Y | N]`#	BES connection holding facility
58	`[set pd_bes_conn_hold_trn_interval =` *back-end-server-connection-hold-time* `]`#	
59	`pdwork -v "`*HiRDB-file-system-area-name*`"[,"`*HiRDB-file-system-area-name*`"]...`	Work table files
60	`{{pdlogadfg -d sys -g` *file-group-name* `[ONL]}}`	System log file configuration
61	`{{pdlogadpf -d sys -g` *file-group-name* `-a "`*system-log-file-name*`" [-b "`*system-log-file-name*`"]}}`	
62	`{{pdlogadfg -d spd -g` *file-group-name* `[ONL]}}`	Synchronization point dump file configuration
63	`{{pdlogadpf -d spd -g` *file-group-name* `-a "`*synchronization-point-dump-file-name*`" [-b "`*synchronization-point-dump-file-name*`"]}}`	
64	`{{[pdplgprm -n` *plug-in-name* `[-s` *shared-memory-size* `]]}}`	Plug-ins

#: If this operand is omitted, the value specified in the same operand in the server common definition is used.

8.2 Operand explanations

8.2.1 Operands related to processes

1) pd_max_bes_process = *maximum-number-of-activated-processes-per-back-end-server*

~<unsigned integer>((1-2048))

Specifies the maximum number of processes that can be activated per back-end server. For multi front-end servers, processes that exceed the value specified in the `pd_max_users` can sometimes become concentrated in a single back-end server. The `pd_max_bes_process` operand specifies the maximum number of processes that can be activated per back-end server when that number exceeds the value of the `pd_max_users` operand.

Condition

This operand is applicable when multiple front-end servers are used.

Specification guidelines

- The value determined by the following formula indicates the maximum number of processes that might possibly become concentrated in a single back-end server.

  pd_max_users value × *number of front-end servers*

  Specify a value for the `pd_max_bes_process` operand by using the value determined here as the upper limit and taking the degree of process concentration in a single back-end server into consideration. Specifying an unnecessarily large value might cause memory shortage.
- If the value specified is smaller than the `pd_max_users` value, the `pd_max_users` value is assumed as the default, and a warning message (`KFPS01888-W`) is output.
- When more processing requests have been issued than there are back-end server processes that can be started, time will be required to process connection requests from the front-end server to the back-end server.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition takes effect. When the same operand is also omitted in the server common definition, the `pd_max_users` value is assumed.

Tuning the specified value

For details about how to tune the maximum number of processes that can be activated, see the *HiRDB Version 9 System Operation Guide*.

2) pd_process_count = *resident-processes-count*[,*resident-processes-count-at-server-startup*]

~<unsigned integer>((0-2048))

resident-processes-count

Specifies the number of processes that can be made resident in the back-end server. A resident process is a process that is activated at the time the server is started.

Advantage

By activating the processes used by transactions that can be processed concurrently by the back-end server at the time of system startup and keeping them resident, the process startup time can be reduced even when new transactions are entered. However, HiRDB startup will take longer.

Specification guidelines

- The value to be specified is determined on the basis of the process private area of the back-end server's server process and the real memory size of the processor. For details about the process private area of server processes, see the *HiRDB Version 9 System Operation Guide*.
- If a multi front-end server configuration is used and if the `pd_max_bes_process` or `pd_max_dic_process` operand is specified in addition, specify for the `pd_process_count` operand a value that satisfies the following conditions:

  pd_process_count value ≤ (*pd_max_bes_process value or pd_max_dic_process value*)
- The value specified in this operand must be no more than the maximum number of processes that can be activated for the back-end server (*pd_max_bes_process value*#).

#

If the pd_max_dic_process or pd_max_bes_process operand is omitted, the default is the pd_max_users value.

Tuning the specified value

For details about how to tune the resident processes count, see the *HiRDB Version 9 System Operation Guide*.

Notes

- Because the number of resident processes has a direct effect on the availability of memory space and on the CPU, specifying an unnecessarily large number might prevent HiRDB from starting or might degrade the server machine's processing performance.
- If more processes than the resident process count are needed, additional processes are dynamically started, up to the maximum processes count allowed. However, depending on the value specified for the pd_max_server_process operand, it might not be possible to start all of the processes indicated by the maximum processes count.

Operand default

When this operand is omitted (or 0 is specified), the specification of the same operand in the server common definition takes effect. When the same operand is also omitted in the server common definition, the maximum number of processes that can be activated is assumed.

resident-processes-count-at-server-startup

Specifies the number of processes that can be made resident during HiRDB startup.

It is common for resident processes to be activated during HiRDB startup. If there are many resident processes, the amount of time required for HiRDB startup increases proportionately. For example, it takes approximately 1 second to activate a process on a server machine with a 100-MIPS performance rating.

The differences in processing that result depending on whether a resident processes count at server startup is specified are as follows:

- When there is no specification of a resident processes count at server startup (when, for example, pd_process_count = 500 is specified)

  All 500 resident processes are activated during HiRDB startup, and HiRDB will not start until they are all activated. In this example, it would take a 100-MIPS server machine about 500 seconds to activate all the resident processes during HiRDB startup.
- When a resident processes count at server startup is specified (when, for example, pd_process_count = 500, 50 is specified)

  Some of the resident processes (50 in this case) are activated during HiRDB startup, and the remaining resident processes are activated after HiRDB startup. HiRDB can be started as long as the specified number of resident processes is activated. In this example, it would take a 100-MIPS server machine about 50 seconds to activate 50 resident processes during HiRDB startup. The remaining 450 resident processes (in this example) would be activated after HiRDB startup.

Advantage

The amount of time required for HiRDB startup is reduced. Use this option when you want to reduce the HiRDB startup time as much as possible, such as when you are using the system switchover facility.

Specification guideline

Specify a value equal to the number of processes that will be required immediately after HiRDB startup is completed.

Notes

When you specify a resident processes count at server startup, recheck the value in the PDCWAITTIME operand of the client environment definition.

If more UAPs than the resident processes count at server startup are to be executed immediately following HiRDB startup, transaction processing might not be performed until after the remaining resident processes have been activated. Therefore, if the value specified in the PDCWAITTIME operand of the client environment definition is small, it might not be possible to process some UAPs due to timeouts. For details about the PDCWAITTIME operand, see the *HiRDB Version 9 UAP Development Guide*.

3) pd_server_cleanup_interval = *interval-for-stopping-nonresident-server-processes*

~<unsigned integer> ((0-1440)) (minutes)

Specifies in minutes the interval for checking for nonresident server processes in HiRDB that are to be stopped. The facility for stopping nonresident server processes is applied when the number of executing server processes exceeds the number of processes that can be made resident (value specified by the `pd_process_count` operand). The number of server processes that the facility stops is computed automatically by HiRDB.

Advantage

This facility improves the utilization rates of the memory and of process resources because it increases the number of nonresident processes that can be reused when the workload (number of server processes that are executing) peaks.

Specification guidelines

- For example, if there is only one hour a day during which peak workload occurs and the intervals between peaks within that hour are approximately two minutes apart, specify `2` for this operand.
- This facility does not have any effect if the number of server processes that execute concurrently during peak periods is always fewer than the number of resident processes. In this case, omit this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Tuning the specified value

Collect statistical information on system operations for each server for one week. Determine the workload peaks from the number of server processes being serviced (`# OF PROCESSES ON SERVICE`). If that peak value exceeds the currently set number of processes that can be made resident (value specified by the `pd_process_count` operand), determine the interval between individual peaks and set that number of minutes.

However, if there are ample resources, such as memory and CPU, in the server machine, adding the shortfall in the number of processes to the number of resident processes (in other words, increasing the value of the `pd_process_count` operand) is more effective in improving performance than specifying the `pd_server_cleanup_interval` operand.

Note

When this operand is omitted or `0` is specified, the system checks every 10 seconds for nonresident server processes that are waiting to be serviced and stops any such processes that are found.

4) pd_max_ard_process = *asynchronous-READ-process-count*

~<unsigned integer>((0-256))

Specify this operand if you use the asynchronous READ facility. For this operand, specify the number of processes necessary for asynchronous READ operations. For a HiRDB/Parallel Server, the value specified here indicates the number of processes per server (back-end server or dictionary server). For details about the asynchronous READ facility, see the *HiRDB Version 9 Installation and Design Guide*.

Condition

A value of `1` or greater must be specified for the `-m` option of the `pdbuffer` operand.

Specification guidelines

- Specify `0` or `1`. However, if a value between 2 and 256 is specified for the `-m` option of the `pdbuffer` operand, specify the same value as the `-m` option value. If a value greater than 256 is specified for the `-m` option, specify the same value as the number of disk devices that store RDAREAs and system files (the number of such disk devices per server for a HiRDB/Parallel Server) or `256`.
- Increasing the value of this operand can shorten the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. Decreasing the value of this operand might increase the processing time when the degree of concurrency is high for the SQL statements to which the asynchronous READ facility is applied. This is because asynchronous READ processes might have to wait for processing completion.
- Because a number of processes equaling *value of this operand* × *server count* are started, determine a value for this operand by taking resources (shared memory and message queue) into consideration. For details about estimating shared memory and message queue sizes, see the *HiRDB Version 9 Installation and Design Guide*.

Tuning the specified value

For details about how to tune the specification value (the number of asynchronous READ processes), see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Operand rule

If you specify `0` for this operand, the asynchronous READ facility is not used.

Relationship to other operands

If you change the value of this operand, re-evaluate the value of the `pd_max_server_process` operand.

Effects on individual estimation formulas

If the value of the `pd_max_ard_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Parallel Server*
- *Formula for size of shared memory used by global buffers* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 2* under *Formulas for the size of the shared memory used by a back-end server*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

5) pd_dfw_awt_process = *number-of-parallel-write-processes-for-deferred-write-processing*

~<unsigned integer>((2-255))

Specify this operand when you use the *facility for parallel writes in deferred write processing* for all buffer pools. Specify for this operand the number of processes to be processed in parallel. Increasing the number of processes can shorten the write processing time. For details about the *facility for parallel writes in deferred write processing,* see the *HiRDB Version 9 Installation and Design Guide*.

Specification guidelines

Specify `2`, which is the smallest value that enables the facility for parallel writes in deferred write processing. Furthermore, to determine the value for this operand, see *Tuning deferred write processing* in the *HiRDB Version 9 System Operation Guide*.

Note

Specifying the facility for parallel writes in deferred write processing increases the number of processes, and consequently raises the CPU usage rate.

Effects on individual estimation formulas

If the value of the `pd_dfw_awt_process` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Parallel Server*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Formula 5* under *Formulas for the size of the shared memory used by a back-end server*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

8.2.2 Operands related to work tables

6) pd_work_buff_mode = each | pool

Specifies the method of allocating buffers when HiRDB creates tables.

`each`: Allocate a buffer for each work table.

`pool`: Allocate a buffer pool for each server process.

Specification guidelines

- Normally, `pool` is specified. `pool` is the appropriate specification when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.
- When the size of the process private area that can be used for work table buffers is predetermined, specify `pool`. When `pool` is specified, HiRDB efficiently allocates work table buffers to work tables.

  In such a case, the process private area is occupied on the basis of the value specified in `pd_work_buff_size`, and input/output operations on work tables are buffered in that pool. Therefore, the process private memory is occupied only to the extent of the value specified in `pd_work_buff_size`.

Notes

If `each` is specified for this operand, the amount of memory used might increase.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `pool` (if `v6compatible` is specified in the `pd_sysdef_default_option` operand, `each` is assumed).

Effects on individual estimation formulas

If the value of the `pd_work_buff_mode` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Considerations when migrating to 64-bit mode*
- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

7) pd_work_buff_size = *work-table-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

Specifies in kilobytes the size of buffers for work tables to be created by HiRDB.

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
Advantage	A large work table buffer size reduces the number of I/O operations associated with data manipulation, which means that the execution time of SQL statements that use work tables is also reduced. However, because each server's process private memory is used, you must also take into account the overall size of the system memory (real memory and virtual memory) when you specify this option. If `pd_work_buff_mode = each` is specified, the memory size to be allocated is *value of pd_work_buff_size* × *required number of work tables*. Therefore, specifying an unnecessarily large value might cause a virtual memory shortage for other processes.	
Application criterion	Specify `pd_work_buff_mode = pool` when a large volume of data is to be retrieved and when manipulations such as `join`, `ORDER BY`, and `GROUP BY` are to be performed.	
Specification guidelines	• Specify the size of the buffer to be allocated for one work table. • If a value greater than the work table memory capacity is specified for the work table buffer size, input/output to the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the work table memory capacity: *Work table memory capacity = Applicable work table size*# ÷ 2	• Specify the size of the buffer pool to be allocated for the entire server process. • Specify a value between `4352` and `5120` when a large volume of data is to be retrieved or when manipulations such as join, `ORDER BY`, and `GROUP BY` are to be performed. Specifying such a value increases the unit of sorting input/output, thus reducing the sort time. • If a large value is specified for the work table buffer size, with the total work table memory capacity for each SQL statement as the upper limit, input/output operations on the work table file during work table creation is eliminated, thus reducing the time necessary for work tables. The following formula can be used to determine the total work table memory capacity for each SQL statement:

Item	pd_work_buff_mode=each is specified	pd_work_buff_mode=pool is specified
		Total work table memory capacity per SQL statement = $a \times b + c \times d$
Notes	• When multiple users execute processes concurrently or when an SQL statement that uses multiple work tables is executed, a buffer of the specified size is allocated for each work table. Consequently, specifying a large value might result in a memory shortage. • If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.	• If the value specified for the work table buffer size is smaller than the number of work tables to be used by each SQL statement, the processing time might become longer than when `each` is specified. Specifically, specify a value that is at least equal to *maximum number of work tables for each SQL statement* × 128. The following formula can be used to determine the maximum number of work tables for each SQL statement: *Maximum number of work tables for each SQL statement* = $b + d$ • If the specified buffer size is too large to be allocated in the system, the allocation of process private memory fails and the server cannot start up.
Operand rule	Specify a multiple of 128. If the value is not a multiple of 128, it is rounded up to the next multiple of 128.	• Specify a multiple of 128. If the value is not a multiple of 128, it is rounded up to the next multiple of 128. • Specify at least 384. If a value that is smaller than 384 is specified, it is rounded up to 384.
Operand default	If this operand is omitted, the specification for the same operand in the server common definition takes effect. If the same operand is also omitted in the server common definition, `512` is assumed.	If this operand is omitted, the specification for the same operand in the server common definition takes effect. If the same operand is also omitted in the server common definition, one of the following values is assumed, depending on your system: • For 32-bit mode: `1024` • For 64-bit mode: `5120`

a:

$\uparrow$ {Capacity of work table (for storing column information)# (kilobytes) ÷ 2} ÷ 128 $\uparrow$ × 128

b:

Maximum number of work tables (for storing column information)#

c:

$\uparrow$ {Capacity of work table (for storing positional information)# (kilobytes) ÷ 2} ÷ 128 $\uparrow$ × 128

d:

Maximum number of work tables (for storing positional information)#

#: For details about how to determine these values, see the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_work_buff_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

8) pd_work_buff_expand_limit = *work-table-buffer-expansion-limit*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((128-1000000))**
- 64-bit mode: **((128-4000000000))**

The size of the work table buffer to be created by HiRDB is specified by the `pd_work_buff_size` operand. Specify the `pd_work_buff_expand_limit` operand if you want to automatically expand a work table buffer when the space in this buffer becomes insufficient. The work table buffer is expanded up to the size specified by this operand.

For example, when the following values are specified for the operands, a 1,024-kilobyte work table buffer is normally allocated. When this size becomes insufficient, the work table buffer is expanded up to 2,048 kilobytes.

- `pd_work_buff_size = 1024`
- `pd_work_buff_expand_limit = 2048`

HiRDB expands a work table buffer in the following cases:

- The necessary work table buffer cannot be allocated when hash execution is applied to an execution method that uses hash join or subquery hash as the joining method.
- A 128-kilobyte work table buffer allocated to each work table becomes insufficient when multiple work tables are concurrently used.

Condition

The `pd_work_buff_mode` operand must be omitted or `pool` must be specified for it.

Advantage

You can prevent a work table buffer shortage (too small a value specified for the `pd_work_buff_size` operand) from causing UAP errors.

Notes

- A work table buffer is not expanded when either of the following conditions is satisfied:
  - The `pd_work_buff_expand_limit` operand is not specified.
  - `pd_work_buff_expand_limit` operand value ≤ `pd_work_buff_size` operand value
- If the specified buffer size is too large to be allocated in the system, the server can still start up, but the allocation of process private memory fails when work tables are created, resulting in an SQL error.

Operand rule

Specify a multiple of 128. If a value other than a multiple of 128 is specified, it is automatically rounded up to a multiple of 128.

Relationship to other operands

When a work table buffer is expanded for the first time in a back-end server process, the `KFPH29008-I` message is output. Note that you can use the `pd_work_table_option` operand to suppress this message output.

Note

After a work table buffer has been expanded, when the number of work tables being used by the applicable server process goes to zero, the expanded work table buffer is released. The number of work tables being used can go to zero in the following cases:

- All cursors that were being used are closed. (In this case, the number of work tables being used might not go to zero.)
- A transaction is normally terminated or cancelled when a holdable cursor is not being used.
- A UAP is disconnected from HiRDB when a holdable cursor is being used.

Remarks

Hash join, subquery hash execution is applied in the following cases:

- *Application of optimizing mode 2 based on cost* and *hash join, subquery hash execution* are specified in the `pd_additional_optimize_level` operand, the `PDADDITIONALOPTLVL` operand of the client environment definition, or the `ADD OPTIMIZE LEVEL` operand of the SQL compile option.
- `HASH` is specified for the SQL optimization specification of the joining method inside an SQL statement.
- `HASH` is specified for the SQL optimization specification of the subquery execution method inside an SQL statement.

Effects on individual estimation formulas

If the value of the `pd_work_buff_expand_limit` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

8.2.3 Operands related to system monitoring

9) pd_spd_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps*

~<unsigned integer>((0, 2-100000))

If an event such as an endless loop occurs in a UAP, acquisition of synchronization point dumps might be skipped. If several synchronization point dumps are not acquired (instead, they are skipped), there will be an increase in the number of overwrite-disabled system log files, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

If HiRDB is forcibly terminated or terminates abnormally when the number of system log files that cannot be overwritten has reached one-half or more of all system log files, a shortage of system log files occurs during rollback processing when HiRDB is restarted. In this case, HiRDB cannot be restarted unless new system log files are added. Any such restart processing will take a longer time than normal.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction). When the number of skipped synchronization point dumps reaches the specified operand value, the target transaction is cancelled forcibly and rolled back. This is called the *skipped effective synchronization point dump monitoring facility*.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

Advantage

Specifying this operand provides a means of dealing with endless loops in UAPs.

Specification guidelines

- Normally, specify `0` for this operand. When `0` is specified, HiRDB computes the upper limit for the skip count. If specifying `0` causes a problem or if the `KFPS02101-I` message is issued, change the value of this operand. For guidelines on the value to specify, see the *HiRDB Version 9 System Operation Guide*.
- If the specified value is too large, all system log files might be placed in overwrite disabled status. If this happens, HiRDB terminates abnormally.
- If the specified value is too small, the number of transactions that are forcibly rolled back might increase.
- Specify this value taking into account the value of `pd_log_sdinterval` and the number of times synchronization point dumps are acquired when the transaction with the longest execution time and the largest log output volume is processed.

Operand default

When this operand is omitted, the value specified for the same operand in the server common definition is assumed. If the same operand is also omitted from the server common definition, the skipped effective synchronization point dump monitoring facility is not used.

Notes

- The monitoring provided for by the specification of this operand begins the first time a synchronization point dump is collected after HiRDB is started (including a restart).
- When this operand is specified, UAPs that are being executed in the no-log mode are also monitored. If the processing of a UAP being executed in the no-log mode is cancelled, the database cannot be automatically recovered, and thus RDAREAs are placed in the error shutdown state. Therefore, when specifying the upper limit, account also for the size of the system logs that are output from other transactions inside the server during the processing of UAPs that are executed in the no-log mode.
- If a transaction is delayed due to a high workload, the skip count increases because a synchronization point dump cannot be acquired until the delayed transaction is completed.
- If the skip count exceeds the upper limit, any process executing a transaction is forcibly terminated. In this case, the rollback logs, which are the logs output by these transactions, are output after the forced termination.
- The `pdload`, `pdmod`, `pdrorg`, `pdexp`, `pddbst`, `pdgetcst`, `pdrbal`, `pdvrup`, `pdmemdb`, and `pdextfunc` commands are not monitored by this facility.

10) pd_dfw_syncpoint_skip_limit = *maximum-number-of-skipped-synchronization-point-dumps-resulting-from-delay-of-synchronization-point-dump-acquisition-due-to-deferred-write-processing*

~<unsigned integer>((0-100000))

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing is completed, acquisition of the synchronization point dump is skipped. This is because acquisition of the synchronization point dump is delayed by the deferred write processing, and the number of update buffers output by the synchronization point exceeds the number of update buffers that can be output within the synchronization point dump acquisition interval.

If more than one synchronization point dump is skipped, there will be an increase in the number of system log files that cannot be overwritten, which might result in a shortage of system log file capacity and ultimately in abnormal termination of the unit.

This operand specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction) due to deferred write processing.

When the number of skipped synchronization point dumps due to deferred write processing reaches the specified operand value, HiRDB determines the maximum number of update buffers in such a manner that acquisition of a synchronization point dump can be completed within the synchronization point dump acquisition interval. If the maximum number of update buffers is exceeded, HiRDB then outputs the oldest update buffer and limits the total size of update buffers at a synchronization point. This is called the *update buffer size restriction facility*.

Advantage

If a synchronization point occurs within the synchronization point dump acquisition interval but before deferred write processing has terminated, the unit terminates abnormally. This operand enables such abnormal termination of the unit to be avoided.

Specification guidelines

Normally, you will omit this operand. If you wish to prevent unit abnormal termination caused by the occurrence of a synchronization point before termination of deferred write processing within the synchronization point dump acquisition interval, specify 1.

If an acceptable number of times synchronization point dumps can be skipped can be determined in advance, such as from the size of the log information, specify that value.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

Operand rules

If 0 is specified in this operand, HiRDB does not use the update buffer size restriction facility.

Notes

The following notes explain the period of effectiveness of the update buffer size restriction facility. The period of effectiveness means the interval between issuance of the `KFPH23035-I` message and issuance of the `KFPH23036-I` message.

- If the number of update buffers exceeds the maximum number of update buffers determined by HiRDB, update buffers are output after update processing has executed; this degrades the update processing throughput. You can use the following formula to obtain the maximum number of update buffers determined by HiRDB:

(*synchronization point dump interval* ÷ *unit value of WRITE*#)

× (1 - (*amount of log information from the previous synchronization point dump to the pre-synchronization point*)

× (*number of buffer sectors in buffer pool* ÷ *total number of buffer sectors in buffer pool that was updated at synchronization point*)

#: For details about the unit value of `WRITE`, see the *HiRDB Version 9 System Operation Guide*.

- If the deferred write trigger is specified in the `pd_dbbuff_rate_updpage` or `pdbuffer -y` operand, and each operand value becomes greater than the maximum number of update buffers determined by HiRDB, the maximum number of update buffers determined by HiRDB is changed to the number of update buffers that triggers deferred write processing.

  The value of the `pdbuffer -w` operand is adjusted automatically so that up to the maximum number of update buffers is output for each buffer.

- A skipped synchronization point dump is detected during update buffer output processing at a synchronization point. Therefore, the update buffer size restriction facility might be enabled after a synchronization point dump is skipped and an error message is displayed.

- Normally, when the parallel writes facility is used, there is one output request per synchronization point for each parallel WRITE process for deferred write processing during synchronization point processing. However, if the update buffer restriction facility is used, there will be more than one output request in order to facilitate early detection of skipped synchronization point dumps.

8.2.4 Operands related to lock

11) pd_lck_pool_size = *server-lock-pool-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-2000000))**
- 64-bit mode: **((1-2000000000))**

Specifies in kilobytes the size of the shared memory area to be used by the back-end server for locking (lock pool).

The area specified by this operand is used for tables for managing locked resources. Note the following relationship between the number of tables for managing locked resources, the number of lock requests, and the lock pool size:

Number of tables for managing locked resources = number of lock requests = size of lock pool × *coefficient*

Specification guidelines

- In the 32-bit mode, the amount of lock pool space required for six lock requests is 1 kilobyte.
- In the 64-bit mode, the amount of lock pool space required for four lock requests is 1 kilobyte.
- The following formulas can be used to determine the value to be specified for this operand:

HiRDB type	Formula (kilobytes)
HiRDB/Parallel Server (32-bit mode)	↑↑ *b* ÷ *value of pd_lck_pool_partition* ↑ ÷ 6 ↑ × *value of pd_lck_pool_partition*
HiRDB/Parallel Server (64-bit mode)	↑↑ *b* ÷ *value of pd_lck_pool_partition* ↑ ÷ 4 ↑ × *value of pd_lck_pool_partition*

b: Total number of transaction lock requests to be executed concurrently by the back-end server. The number of lock requests depends on the SQL. For details about how to determine the total number of lock requests, see *D. Determining the Number of Locked Resources*.

Note:

When you execute `DROP TABLE` or `DROP SCHEMA` in a definition SQL, it is especially important to have already determined in advance an appropriate value for this operand.

Tuning the specified value

See the usage rate for the locked resources management table (`% OF USE LOCK TABLE`) displayed for the back-end server in the statistical information on system operation by the statistics analysis utility. If the maximum usage rate equals or exceeds 80%, it is recommended that the operand's value be increased in preparation for future database expansion. If the maximum usage rate does not exceed 10%, it is recommended that the operand's value be decreased to conserve shared memory space.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the appropriate value shown below:

- For 32-bit mode: `16000`
- For 64-bit mode: `32000`

However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `1024`.

Note

- If the value specified for this operand is too small, an SQL statement might return an error.
- Do not specify a larger value than necessary in this operand. A large value will increase the size of the shared memory used by HiRDB, which might cause a shortage of shared memory and prevent HiRDB from starting.
- If you do an all-item search of a table that contains many rows while locking is in units of rows, this operand's value will need to be increased to reflect the large number of items, which will increase the amount of memory required. Instead, consider making the following adjustments in the UAP:
    - Acquire locks in table units.

- If you can use the unlocked search facility, perform searches unlocked.
- Narrow the search conditions, and divide the processing into several transactions.

Relationship to other operands

This operand is related to the `pd_lck_pool_partition` operand.

12) pd_lck_pool_partition = *per-server-lock-pool-partition-count*

~<unsigned integer>((1-5000))

Specifies the number of lock pool partitions to be used in locking by the back-end server when distributing lock processing.

For details about distributing lock processing, see the *HiRDB Version 9 System Operation Guide*.

Tuning the specified value

For details about how to tune the number of lock pool partitions, see the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `1`.

Notes

- When the value set in this operand is too large, the size of the shared memory assigned to each lock pool partition becomes smaller, which might result in not enough lock pool partition capacity, causing SQL to return an error. Lock processing also takes time, so system performance declines. If this becomes an issue, specify a smaller value in this operand.
- The lock pool size must be at least 1 kilobyte. If a value larger than the value of `pd_lck_pool_size` is specified, the `KFPS00421-W` message will be issued and the value of `pd_lck_pool_size` will be assumed for this operand.

Relationship to other operands

This operand is related to the following operands:

- `pd_lck_pool_size`
- `pd_lck_deadlock_check_interval`

Effects on individual estimation formulas

If the value of the `pd_lck_pool_partition` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Processes started by HiRDB/Parallel Server*
- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

13) pd_lck_until_disconnect_cnt = *total-number-of-tables-and-RDAREAs-to-be-locked-until-disconnect-specification*

~<unsigned integer>((0-140000))

Specifies the number of resources to be locked for the tables and RDAREAs that are to be held across transactions. Based on the value specified for this operand, blocks for which lock with `UNTIL DISCONNECT` is specified for the tables and RDAREAs are allocated in the shared memory.

Specification guidelines

Normally, this operand need not be specified. Specification of a value other than the default value might be necessary in the following cases:

- When the number of utilities to be executed concurrently increases
- When a holdable cursor is used
- When the local buffer specified in the `pdlbuffer` operand is used
- When a shared RDAREA is used
- When an SQL session-specific temporary table is used

For details about how to estimate the specification value for this operand, see *C.5 Formula for determining total number of tables and RDAREAs per server locked with UNTIL DISCONNECT specified (pd_lck_until_disconnect_cnt)*.

Tuning the specified value

If the value specified for this operand is small, a transaction might roll back or a utility might terminate abnormally with return code `8`. In such cases, the message `KFPA11914-E` or `KFPH28001-E` is output. If this occurs, increase the value of this operand.

When the value of this operand is increased, the amount of required memory space increases proportionately. The required memory size can be expressed as follows: *value of this operand* × 48 (64 in the 64-bit mode) bytes.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `256`.

Effects on individual estimation formulas

If the value of the `pd_lck_until_disconnect_cnt` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the number of records in a synchronization point dump file*
- *Formula 2* under *Formulas for the size of the shared memory used by a back-end server*

14) pd_max_open_holdable_cursors = *maximum-number-of-holdable-cursors-that-can-be-concurrently-open-when-LOCK-statement-with-UNTIL-DISCONNECT-specification-is-not-executed*

~<unsigned integer>((16-1024))

When you use holdable cursors for a table for which a `LOCK` statement with `UNTIL DISCONNECT` specification is not executed, this operand specifies the maximum number of holdable cursors that can be concurrently open for each transaction.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `16`.

Note

Specifying a value other than the default value for this operand increases the amount of shared memory used.

Relationship to other operands

The values specified for this operand and the following operands are used for computing the shared memory size for lock servers. For a 32-bit mode HiRDB system, if the values specified for these operands are too large, the shared memory size of the lock servers exceeds 2 GB, and as a result, HiRDB might not start. Therefore, adjust the values specified for these operands so that the shared memory size of the lock servers does not exceed 2 GB.

- `pd_max_access_tables`
- `pd_max_users`
- `pd_max_bes_process`
- `pd_lck_hash_entry`
- `pd_lck_pool_size`

For details about shared memory, see the *HiRDB Version 9 Installation and Design Guide*.

15) pd_lck_hash_entry = *lock-pool-hash-entry-count*

~<unsigned integer> ((0-2147483647))

Specifies the number of hash table entries to be used in the lock pool. According to the value specified here, HiRDB allocates a lock pool in the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand.

Consider specifying a value for this operand in the following cases:

- If you do not wish to change the shared memory size if possible when upgrading to version 06-02 or later, specify `11261`. In this case, the same number of hash entries is allocated as in the earlier version, and the hash table size inside the lock pool remains the same as before.
- It is possible to improve performance by specifying in this operand a value greater than the recommended value shown below. However, specifying a value greater than variable *a* (also shown below) will not improve performance over the case in which *a* is specified.

  The recommended value is as follows:

  Recommended value = Largest prime number not exceeding MAX($\uparrow a \div 10 \uparrow$, 11,261)

Variable	Formula for computing the variable	
a	$(b + 3) \times 10 +$ *pd_lck_pool_size value* $\times\ c$	
b	If `pd_max_users` value > `pd_max_bes_process` value	*pd_max_users value*
	If `pd_max_users` value $\leq$ `pd_max_bes_process` value	*pd_max_bes_process value*
c	6 for the 32-bit mode; 4 for the 64-bit mode	

Operand rules

- If this operand and the `pd_lck_hash_entry` operand of the server common definition are both omitted or `0` is specified in this operand, HiRDB calculates a recommended value for the server. (However, if `v6compatible` has been specified in the `pd_sysdef_default_option` operand, the default for this operand is `11261`.)
- When a value that is neither `0` nor a prime number is specified in this operand, HiRDB assumes that the specification is the largest prime number that does not exceed the specified value.

Note

If the value specified in this operand is too small, there might be an insufficient number of hash entries, and performance might deteriorate. If this operand is omitted, there will never be a shortage of hash entries and performance will not deteriorate due to an insufficient number of hash entries.

16) pd_dbsync_lck_release_count = *global-buffer-lock-release-interval-during-synchronization-point-processing* ~<unsigned integer>((0, 100-1073741824))

Specifies an interval for unlocking global buffers, when global buffer locking occurs during synchronization point processing.

During synchronization point processing, search processing occurs on the buffers (update buffers) that must be applied to the disk. Normally, global buffers are unlocked at a specific interval during search processing on the update buffers.

For example, if 100 is specified in this operand, a global buffer is unlocked once when search processing on 100 sectors (global buffer sectors) is completed. After that, the global buffer is locked again and search processing is resumed. In this example, unlocking occurs once every 100 sectors.

Advantage

By specifying this operand, you can adjust the global buffer lock time during synchronization point processing. When a small value is specified in this operand, the global buffer lock time becomes short and transaction performance might improve during synchronization point processing.

To obtain the global buffer pool lock time, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Buffer pool lock time during synchronization point processing (SYNCL)`.

Specification guidelines

Normally, there is no need to specify this operand. Consider specifying this operand when both the following conditions apply:

- Transaction performance drops during synchronization point processing.
- A large number of buffer sectors is specified in the `-n` option of the `pdbuffer` operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `10000`.

Operand rules

- If the specified value is in the range 1 to 99, 100 is set automatically.
- If 0 is specified, global buffers are locked until update buffer search processing is completed.

Notes

If a small value is specified in this operand, the update buffer search time increases due to interrupts from other transactions and CPU usage rises. The global buffers updated during that time are also output during synchronization point processing. Therefore, the number of update buffers to be output during synchronization point processing increases. To obtain the number of update buffers to be output during synchronization point processing, execute the statistics analysis utility and in the global buffer pool statistical information check the item called `Number of synchronization point output pages (SYNCW)`.

As you increase the value of this operand, the lock time for a global buffer to determine the buffer to be output at a synchronization point will increase. For this reason, lock contention grows substantially during synchronization point processing, which might affect transaction performance.

8.2.5 Operands related to buffers

17) pd_sql_object_cache_size = *SQL-object-buffer-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((22-256000))**
- 64-bit mode: **((22-2000000))**

Specifies in kilobytes the size of the buffer area (shared memory) in which SQL objects are to be placed.

Specification guidelines

- SQL objects are saved in a buffer until the user's transaction has terminated. The buffer must be large enough to store the SQL objects of all transactions that will be executed concurrently.
- If the SQL object buffer hit rate is low, the performance might decreases due to the overhead of SQL statement analysis processing.
- SQL analysis can be reduced by saving the SQL objects of static SQLs in the buffer after transaction termination (until the buffer runs out of space) and sharing them among multiple users who execute the same UAP. To effectively utilize the buffer, allocate it so that the SQL objects of frequently-used UAPs are resident in the buffer.
- To estimate the buffer size, first determine the buffer size needed for UAP execution from the length of the SQL objects from the SQL statements to be issued by the UAP. Then, compute the buffer size by considering the number of UAPs that will be executed concurrently and the number of concurrently executing users.
- For details about how to estimate the length of the SQL object from a single SQL statement, see *C.2 Formulas for determining size of SQL object buffer (pd_sql_object_cache_size).*

Operand default

If this operand is omitted, the value specified for the same operand in the server common definition or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition and the system common definition, the following value is assumed:

(*value of pd_max_users* + 3) × 22

Tuning the specified value

For details about how to tune the SQL object buffer size, see the *HiRDB Version 9 System Operation Guide*.

Effects on individual estimation formulas

If the value of the `pd_sql_object_cache_size` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for the size of the shared memory used by a back-end server*

8.2.6 Operands related to shared memory

18) pd_bes_shmpool_size = *back-end-server-shared-memory-size*

~<unsigned integer> (kilobytes)

- 32-bit mode: **((1-200000))**
- 64-bit mode: **((1-4000000000))**

Specifies the size of the area (in kilobytes) to be used by a back-end server as part of the shared memory for the unit controller.

Specification guidelines

Normally, omit this operand. If this operand is omitted, HiRDB computes a value for it (however, if `v6compatible` is specified for the `pd_sysdef_default_option` operand, `1024` is assumed). HiRDB computes this value based on the values for the related definitions. Therefore, specify the appropriate values for the related definitions. For details about the related definitions, see *Formulas for the size of the shared memory used by a back-end server* in the *HiRDB Version 9 Installation and Design Guide*.

If the values of any of the operands of the variables used in the estimation formula are changed, HiRDB automatically re-calculates this operand value.

Tuning the specified value

If any of the following messages is output, increase the specification value of this operand.

- `KFPA20003-E`
- `KFPD00005-E`
- `KFPD00012-E`
- `KFPD00021-E`
- `KFPH20003-E`

Notes

If this operand is omitted and the appropriate values are not specified in the related definitions, the value obtained by HiRDB might be too large, resulting in the allocation of an unnecessarily large amount of shared memory, or too small, resulting in one or more of the following problems:

- A unit does not start.
- A UAP or utility does not execute.

If the value specified for this operand is unnecessarily large or unnecessarily small, the same problems also occur.

8.2.7 Operands related to the RPC trace information

19) pd_rpc_trace = Y | N

Specifies whether RPC trace information is to be collected. HiRDB maintenance information is output in the RPC trace information.

Normally, omit this operand.

`Y`: Collect RPC trace information.

`N`: Do not collect RPC trace information.

Note

Specifying `Y` for this operand degrades communication performance.

Operand default

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `N` is assumed.

20) pd_rpc_trace_name = "*name-of-RPC-trace-collection-file*"

~<path name of up to 254 characters>

Specifies as an `absolute` path name the file name for the RPC trace files. Three RPC trace files are created, with 1, 2, and *l* suffixed to the specified file name.

Note

Files with a maximum size of *pd_rpc_trace_size value* × 2 are created under the directory specified by this operand. Pay attention to the file size.

Operand default

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, the following value is assumed:

`%PDDIR%\spool\rpctr`

21) pd_rpc_trace_size = *RPC-trace-collection-file-size*

~<unsigned integer> ((1024-2147483648)) (bytes)

Specifies in bytes the size of the RPC trace files.

The value specified by this operand determines the size of both RPC trace file 1 and RPC trace file 2.

Specification guidelines

An RPC trace collects in a single file in time order the information on the communications among all processes. If the volume of this information exceeds the specified value, trace data output is switched to the other file, which means that older trace information will be overwritten. When this happens, there might not be enough trace information available, making troubleshooting difficult. For this reason, specify at least `1000000` for this operand.

Note

The value specified by this operand does not apply to RPC trace file *l*, because its size is fixed at 0 bytes.

Operand default

If this operand is omitted, the value specified for the same operand in the server common definition, the unit control information definition, or the system common definition takes effect, in that order. If the same operand is also omitted from the server common definition, the unit control information definition, and the system common definition, `4096` is assumed.

8.2.8 Operands related to troubleshooting information

22) pd_module_trace_max = *maximum-number-of-module-traces-that-can-be-stored*

~<unsigned integer>((126-16383))

A HiRDB process records the history of the executed functions and macros inside the process private memory. This history is called a *module trace*. This operand specifies the number of module trace records. The content of this history is loaded into the `core` file and is output when a process error occurs.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `126`.

Note

Process private memory of the following size is allocated to each process:

In the 32-bit mode: 64 + 48 × *value of pd_module_trace_max operand* (bytes)

In the 64-bit mode: 64 + 64 × *value of pd_module_trace_max operand* (bytes)

23) pd_module_trace_timer_level = 0 | 10 | 20

Specifies how to acquire the time to be output in module traces. The following table explains the meaning of the value specified for this operand.

Specified value	Time acquisition method
`0`	Time is output in seconds at every module trace output location.
`10`	Time is output in microseconds only at performance-critical module trace output locations, such as those before and after input/output processing, and time is output in seconds at other locations.
`20`	Time is output in microseconds at every module trace output location.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a performance check purpose or the like, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `0`.

Note

If you specify a value other than `0` for this operand, a function for acquiring time in microseconds is issued, and as a result, system performance might decline.

24) pd_pth_trace_max = *maximum-number-of-stored-communication-traces*

~<unsigned integer>((1024-8388608))

Specifies the maximum number of communication trace records to be used as troubleshooting information.

Specification guidelines

Normally, there is no need to specify this operand. If a maintenance engineer asks you to specify this operand for a reason such as performance checking, follow the maintenance engineer's instructions.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the default is `1024`.

Notes

Increasing the value of this operand increases the amount of process private memory secured by HiRDB processes.

Process private memory for communication traces is calculated based on this operand's value rounded up to the power of two. For details about memory requirements, see *Calculation of required memory* in the *HiRDB Version 9 Installation and Design Guide*.

Effects on individual estimation formulas

If the value of the `pd_pth_trace_max` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Calculation of required memory* under *Estimating the memory size required for a HiRDB/Parallel Server*

8.2.9 Operands related to global buffers

25) pd_max_add_dbbuff_no = *maximum-global-buffers-count-for-dynamic-addition*

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, this operand specifies the maximum number of global buffers (per server) that can be added dynamically by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

- Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify a sufficient value based on that value.

- Determine the operand's value in such a manner that the following condition is satisfied:
  Value of `pd_max_add_dbbuff_no` $\leq$ 2,000,000 - number of global buffers allocated per server during HiRDB startup

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the appropriate value shown below:

Condition		Operand default
32-bit mode	$a \geq 500$	256
	$a < 500$	500 - a
64-bit mode	$a \geq 1,000$	256
	$a < 1,000$	1,000 - a

a: Number of global buffers allocated per server during HiRDB startup

Notes

Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Determining the value of S* under *Determining the size of status files*
- *Formula 2*, *Formula 4*, *Formula 5*, and *Formula 6* under *Formulas for the size of the shared memory used by a back-end server*

26) pd_max_add_dbbuff_shm_no = *maximum-shared-memory-segments-count-for-dynamic-addition*

~<unsigned integer>((1-32752))

In order to change global buffers dynamically, this operand specifies the maximum number of shared memory segments (per server) that can be allocated when dynamic addition is performed by the `pdbufmod` command.

Condition

`Y` must be specified in the `pd_dbbuff_modify` operand.

Specification guidelines

Estimate the number of global buffers to be added dynamically by the `pdbufmod` command and then specify an appropriate value.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the appropriate value shown below:

Condition		Default value
`pd_max_add_dbbuff_no` operand is omitted.	32-bit mode	500 + A
	64-bit mode	1,000 + A

Condition		Default value
`pd_max_add_dbbuff_no` operand is specified.	32-bit mode	↓ *value of pd_max_add_dbbuff_no* × 1.5 + *A* ↓ (If the value is 32,752 or greater, `32752` is set)
	64-bit mode	

A: Remaining number of shared memory segments that can be allocated during HiRDB startup. This value can be obtained from the following formula:

A = *a* - *b*

Assign the following values to *a* and *b*:

a = *value of pd_max_dbbuff_shm_no* (in 64-bit mode, 16)

b = *number of shared memory segments allocated to each server during HiRDB startup*

You can obtain information about the shared memory segments by using the `pdls -d mem` command or an OS command.

Notes

- If the following condition is satisfied, the value of the `pd_max_add_dbbuff_no` operand is assumed in this operand:

  Value of `pd_max_add_dbbuff_shm_no` < value of `pd_max_add_dbbuff_no`

  The value of the `pd_max_add_dbbuff_no` operand is also assumed when the default value satisfies the above condition.
- Do not specify an unnecessarily large value in this operand. If this operand's value is too large, the shared memory used by HiRDB increases, which might result in a shortage of shared memory and an inability of HiRDB to start.
- If the size of a shared memory segment to be added exceeds the `SHMMAX` operand value, shared memory is divided into multiple segments based on the `SHMMAX` operand value as the maximum value. Either increase the `SHMMAX` operand value based on the size of the shared memory segment to be added or increase the `pd_max_add_dbbuff_shm_no` operand value so that no shortage occurs when the shared memory is segmented.
- If the facility for dynamically changing global buffers is used, the total number of shared memory segments that are allocated for the global buffers can be obtained from the following formula:

  pd_max_dbbuff_shm_no value + *pd_max_add_dbbuff_shm_no value*

  Therefore, if more shared memory segments were allocated when HiRDB started than the number specified for `pd_max_dbbuff_shm_no`, the number of shared memory segments that are actually allocated during dynamic change is the value of `pd_max_add_dbbuff_shm_no` minus the excess number of shared memory segments.

Relationship to other operands

This operand is related to the following operands:

- `SHMMAX`
- `pdbuffer`
- `pd_max_add_dbbuff_no`
- `pd_max_dbbuff_shm_no`

Effects on individual estimation formulas

If the value of the `pd_max_add_dbbuff_shm_no` operand is changed, the following estimation formulas are affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*
- *Determining the value of S* under *Determining the size of status files*
- *HiRDB/Parallel Server* under *Determining Environment Variables Related to the Number of Resources*

8.2.10 Operands related to temporary tables

27) pd_max_temporary_object_no = *maximum-number-of-temporary-tables-and-temporary-table-indexes-used-at-any-one-time*

~<unsigned integer> ((0-131072))

Specifies the maximum number of temporary tables and temporary table indexes that can be used at any one time for each server.

Specification guidelines

Use the formula shown below to determine the value of this operand. For a HiRDB/Parallel Server, obtain the value for each back-end server.

Maximum value of (*the number of transaction-specific temporary tables*
+ *total number of temporary table indexes for those temporary tables*) × *number of activities*#
+ (*number of SQL session-specific temporary tables used in the SQL session*
+ *total number of temporary table indexes for those temporary tables*) × *number of connected users*

#

number of activities:

((*value of pd_max_users* + 3) × 2 + 1) + α

α : If the value specified for `pd_max_users` is `60` or less, 5; if it is `61` or greater, 0.

For a HiRDB/Parallel Server, this operand value is applied to each back-end server. Therefore, as a guideline, specify the largest value used among all back-end servers in this operand

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition or the system common definition is used, in that order. If the same operand is omitted from the server common definition and the system common definition, `0` is assumed.

Notes

The value of this operand affects the size of shared memory used by HiRDB; therefore, do not specify a value that is greater than the value described in *Specification guidelines*. If the specified value is greater than the value described in *Specification guidelines*, HiRDB might not be able to start due to a shortage of shared memory.

Relationship to other operands

This operand is related to the `pd_max_tmp_table_rdarea_no` operand.

Effects on individual estimation formulas

If the value of the `pd_max_temporary_object_no` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 8* under *Formulas for the size of the shared memory used by a back-end server*

8.2.11 Operands related to delayed batch creation of plug-in index

28) pd_plugin_ixmk_dir = "*index-information-file-creation-directory-name*" or "*index-information-file-creation-HiRDB-file-system-area-name*"

~<path name>

Specifies the name of the directory under which the index information file for delayed batch creation of a plug-in index is to be created. Specify a HiRDB file system area name in order to create the index information file in a HiRDB file system area. An absolute path name must be used for the directory name or HiRDB file system area name.

For details about delayed batch creation of a plug-in index, see the *HiRDB Version 9 System Operation Guide*.

Notes

- The directory (or HiRDB file system area) specified here must have been created in advance. If a nonexistent directory (or HiRDB file system area) is specified, an error will result during execution of a

UAP that specifies delayed batch creation of plug-in indexes (UAP that is executed in an environment in which `PDPLGIXMK = YES` is specified in the client environment definition).

- Once the UAP has executed, the value specified for this operand must not be changed before delayed batch creation of plug-in indexes is performed by the database reorganization utility. If it is changed, a plug-in index delayed batch creation cannot be performed.

8.2.12 Operands related to Java

Java operands are specified when a Java stored procedure or Java stored function is used. For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide*.

29) pd_java_stdout_file = *"Java-virtual-machine-standard-output-and-standard-error-output-destination-file"*
~<path name>

Specifies as an absolute path name the file to which the standard output and standard error output are to be output in a Java virtual machine.

Specification guideline

Because the size of the file specified by this operand is extremely large, this operand is not normally specified. It is recommended that this operand be specified during debugging of a Java stored procedure or Java stored function. There is no limit to the size of the file that can be specified by this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition, the unit control information definition, or the system common definition, in that order, is assumed. When the same operand is also omitted in the server common definition, the unit control information definition, and the system common definition, the Java Virtual Machine standard output and standard error output are ignored.

Note

If there are simultaneous writing attempts from multiple processes, their output contents cannot be guaranteed.

Operand rules

- Up to 255 characters can be used for the path name.
- Path names are not case sensitive.

30) pd_java_castoff = Y | N

Specifies whether to use the following events as triggers for shutting down the process at the server (single server, front-end server, dictionary server, or back-end server) that started the Java Virtual Machine:

No.	Server type	Process name	Trigger that ends process
1	Single server	`pdsds`	UAP is disconnected
2	Front-end server	`pdfes`	UAP is disconnected
3	Dictionary server	`pddic`	Transaction is completed or UAP is disconnected
4	Back-end server	`pdbes`	Transaction is completed or UAP is disconnected

`Y`: Shut down server process when trigger occurs.

`N`: Do not shut down server process when trigger occurs.

Specification guidelines

Normally, this operand is not specified. However, if you encounter the problems described below, we recommend that you specify `Y` in this operand:

- Use of the Java Virtual Machine causes the amount of memory usage to increase to the point where the available system memory becomes nearly exhausted.
- SQL code that includes numerous search conditions is executed, and even though the connection does not use the Java Virtual Machine, the maximum stack size set by the Java Virtual Machine on another connection prevents the stack from expanding, causing the server process to be aborted by a segmentation error.

For details about the Java Virtual Machine facility, see the Java Virtual Machine documentation.

Notes

On systems that run Java stored routines frequently, specifying `Y` in this operand will generate overhead for server process restarts and Java Virtual Machine startups.

Relationship to other operands

This operand is related to the `pd_process_count` operand.

8.2.13 Operands related to system log files

31) pd_log_dual = Y | N

Specifies whether dual system log files are to be used.

`Y`: Use dual system log files.

`N`: Do not use dual system log files.

Advantages

When dual system log files are used, HiRDB collects the same system log information in both files, which are called File A and File B. If a failure occurs in one of the files while the collected system log file is being loaded, the file can be loaded from the other file, resulting in higher system reliability.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

When use of dual system log files is specified, the name of the File B system log file must be specified with the `pdlogadpf` operand.

32) pd_log_remain_space_check = warn | safe

Specifies the processing to be performed by HiRDB when the available space in the system log file falls below the warning level. This is called the facility for monitoring the free area for the system log file. For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

`warn`:

When the available space in the system log file falls below the warning level, the `KFPS01162-W` message is output.

`safe`:

When the available space in the system log file falls below the warning level, scheduling of new transactions is suppressed, all transactions inside the server are forcibly terminated, and the `KFPS01160-E` message is output.

Specification guideline

We recommend that you specify `safe` because it can reduce the probability of abnormal termination of units due to system log file space shortage. However, when `safe` is specified, all transactions inside the server are forcibly terminated when a shortage occurs in the available space in the system log file. Therefore, the design of the system log file requires more accuracy. For details about system log file design, see the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `warn`.

33) pd_log_auto_unload_path = "*unload-log-file-output-directory*"[,"*unload-log-file-output-directory*"]...

~<path name>((1-136 characters))

Specifies as absolute path names the unload file output directories when the automatic log unloading facility is to be used for the system log. A HiRDB file system area name must be specified to create the unload log file in a HiRDB file system area. *The directories or HiRDB file system areas specified for this operand must be created before HiRDB is started.*

Additionally, in this operand, specify a different directory or HiRDB file system area for each server.

For details about the automatic log unloading facility, see the *HiRDB Version 9 System Operation Guide*.

Specification guidelines

It is important to check the available disk space before specifying a directory, so as to ensure that the created unload log file does not cause a disk space shortage.

If an unload log file cannot be created in the specified directory because of a disk space shortage, the automatic log unloading facility stops. If this is a possibility, creation of multiple directories is recommended.

Note, however, that the database recovery operation of selecting the unload files needed for recovery is simplified somewhat when only one directory is used.

Also keep in mind the following when multiple directories are created:

- It is recommended that directories be specified in different partitions to protect against disk errors.
- If the unload log file cannot be created in a single directory because of a full disk or disk error, create an unload log file under a different directory. HiRDB uses the directories specified by this operand in the order of their specification.

Operand rules

- Up to 128 directories can be specified.
- When multiple directories are specified, the same path name cannot be specified.

Notes

- The automatic log unloading facility cannot be used in the following cases:
  - `N` is specified in the `pd_log_unload_check` operand.
- If two or more directories are created for unload log files and there is no empty directory when HiRDB starts normally, the automatic log unloading facility stops.
- When a multi-HiRDB is being used, you must create a different directory for each HiRDB. Specifying the same directory for more than one HiRDB will make it impossible to determine which unload log file applies to which HiRDB.

34) pd_log_auto_unload_restart = Y | N

Specifies whether the automatic log unloading facility is to be restarted when the system log files are swapped following the issuance of a message indicating that the automatic log unloading facility stopped due to an error (message `KFPS01150-E`).

`Y`: Restart the automatic log unloading facility.

`N`: Do not restart the automatic log unloading facility.

Condition

The following two conditions must be satisfied:

- `Y` is specified in the `pd_log_unload_check` operand or this operand is omitted.
- The `pd_log_auto_unload_path` operand is specified.

Advantages

- If `Y` is specified in this operand and a shortage of disk capacity occurs due to an increase the number of work files, or because the unload processing fails due to a temporary error such as a process creation error, HiRDB automatically restarts the unloading of system logs the next time the system log files are swapped.
- If `N` is specified in this operand and the unload log files are set to be saved while the automatic log unloading facility is stopped due to an error, you can prevent the unload log files from being saved before the `pdlogatul -t` command is executed.

Specification guidelines

Normally, specify `Y` or omit this operand.

Specify `N` if you have already set HiRDB to monitor the automatic log unloading facility termination message (`KFPS01150-E` message) and restart the facility with the `pdlogatul -b` command.

Operand default value

If this operand is omitted, the value specified for the same operand in the server common definition takes effect. If the same operand is also omitted from the server common definition, `Y` is assumed.

35) pd_log_singleoperation = Y | N

This operand is applicable when dual system log files are used; it need not be specified when dual system log files are not used.

Specifies whether the single-operation mode is to be used for the system log files. Even if an error occurs in a system log file, making no dual system log files available, HiRDB (or a unit for a HiRDB/Parallel Server) can continue processing using the remaining single normal system log file without being abnormally terminated. This is called *single operation of the system log files*.

The mode in which continuation of processing is permitted only with both system log files available (normal operation mode) is called *double operation of the system log files*.

`Y`: Use single operation of the system log files.

`N`: Do not use single operation of the system log files. Both system log files must always be used.

Condition

This operand is valid only when `pd_log_dual = Y` is specified.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

36) pd_log_rerun_reserved_file_open = Y | N

Specifies whether a system log file is to be opened automatically.

If no system log file that can be overwritten is available during a unit restart, HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of system log file*.

A reserved file is used in the following cases:

- Between the time of a restart and the time when the first synchronization point dump is collected
- When none of the opened file groups can be overwritten

`Y`: Open a system log file automatically (open and use a reserved file).

`N`: Do not open a system log file automatically (do not use a reserved file).

Advantages

When `Y` is specified, the unit can be restarted as long as a reserved file is available, even though no file that can be swapped in is available during the unit restart.

However, if a file that is in unload wait status is available, the unit is stopped; once that file has been unloaded, the unit can be restarted.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

37) pd_log_rerun_swap = Y | N

Specifies whether the system log files are to be swapped during a unit restart.

`Y`: Swap the system log files.

`N`: Do not swap the system log files.

Advantage

When `Y` is specified, there can be a physical separation between the system log files used before and after a restart. Therefore, the system log file that was being used before the restart can be reused during server operation.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

38) pd_log_swap_timeout = *wait-time-for-completion-of-system-log-file-swapping*

~<unsigned integer>((1-32580)) (seconds)

Specifies in seconds the wait time during which swapping of system log files must be completed. If a swap of system log files is not completed within the specified amount of time, the unit terminates abnormally.

Specification guidelines

Normally, there is no need to specify this operand. If it takes a long time to swap system log files for a reason such as poor machine performance, you can specify this operand with a value that is greater than the default value. To detect errors or delays quickly during system log file swapping so that the unit can be terminated (such as because of a disk failure), reduce the value of this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `180`.

39) pd_log_unload_check = Y | N

Specifies whether HiRDB is to check the unload status of system log files.

`Y`:

Check the unload status (normal operation).

`N`:

Do not check the unload status. A system log file is placed in swappable status, regardless of its unload status, when both of the following conditions are satisfied:

- It is in overwritable status
- It is in extraction completed status (HiRDB Datareplicator)

In such a case, the system log file operation method is to release checking of unload status. For details about this operation method, see the *HiRDB Version 9 System Operation Guide*.

Advantages

Specifying `N` provides the following advantages:

- Operations are simplified because the system log file unload operation is eliminated.
- It is not necessary to provide files for storing unload files.

Specification guideline

Specify `N` if the system log file will not be needed for database recovery (in other words, if recovery from a backup collection point will be sufficient).

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `Y`.

Notes

The following points apply when `N` is specified:

- Database can be recovered only if backups have been made.
- If this option is specified when the system log file is required for database recovery, it will not be possible to recover the database.

40) pd_log_max_data_size = *log-input/output-buffer-size*

~<unsigned integer>((32000-523000)) (bytes)

Specifies in bytes the size of the buffer to be used for system log input/output operations.

Specification guideline

Specify a value that satisfies conditional expression 1 below. If `uap` is specified in the `pd_rpl_reflect_mode` operand or a recovery-unnecessary front-end server is used, specify a value that satisfies both conditional expressions below (1 and 2). To optimize the value, use the tuning method for the specification value.

Conditional expression 1: log input/output buffer length $\geq$ *a*

a: 72 $\times$ (*maximum number of back-end and dictionary servers that are targets for reference or update processing by a single transaction*) + 1,344

Conditional expression 2: log input/output buffer length $\geq$ *b*

b: (*maximum number of back-end and dictionary servers that are targets for reference or update processing by a single transaction* + 1) $\times$ 128 + 64

Tuning the specified value

A value (other than the default value) might need to be specified for this operand after the following types of statistics analysis utility statistical information related to system operation have been checked:

- Number of buffer sectors waiting for input/output (`# OF BUFFER FOR WAIT I/O`)

  If the average number of buffer sectors waiting for input/output significantly exceeds 100, increase the value for this operand so that the average approaches 100.
- Number of waits caused by lack of a current buffer (`# OF WAIT THREAD`)

If the number of waits caused by lack of a current buffer is not 0, increase the value for this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `400000`. (However, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `32000`.)

Notes

The specification of this operand affects the response and throughput of SQL code executed by a transaction. When a small value is specified, writing to system log files occurs frequently, which might result in deterioration of performance.

Relationship to other operands

Use this operand and the `pd_log_write_buff_count` operand to determine the log I/O buffer size.

41) pd_log_write_buff_count = *log-output-buffer-sectors-count*

~<unsigned integer>((3-65000))

Specifies the number of buffer sectors to be used for system log output.

Tuning the specified value

Specify `10` (the default) for this operand initially. Thereafter, use a statistics analysis utility to obtain statistical information related to system operation to check the number of waits caused by a shortage of current buffer space (`# OF WAIT THREAD`). If the number of waits is high, specify a larger value in order to improve throughput.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `10`. (However, if `v6compatible` or `v7compatible` has been specified in the `pd_sysdef_default_option` operand, the default is `3`.)

Notes

If a small value is specified in this operand and the number of transactions is high, multiple transactions might be waiting for system log output, lowering system performance.

Relationship to other operands

Use this operand and the `pd_log_max_data_size` operand to determine the log output buffer sector count.

42) pd_log_rec_leng = *system-log-file-record-length*

~<unsigned integer>((1024, 2048, 4096)) (bytes)

Specifies the record length for the system log files; the specifiable values are `1024`, `2048`, and `4096`.

Specify the record length specified in the `-l` option of the `pdloginit` command for this operand.

Specification guidelines

Specify the record length based on the guidelines for designing the record length of system log files. For details about the guidelines for designing the record length of system log files, see *Record length of a system log file* in the *HiRDB Version 9 Installation and Design Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `4096`.

Notes

- If a value that is different from the record length specified by the `-l` option of the `pdloginit` command is specified for this operand, system log files cannot be opened.
- For details about how to modify the system log file record length, see the *HiRDB Version 9 System Operation Guide*.

43) pd_log_rollback_buff_count = *rollback-log-input-buffer-sector-count*

~<unsigned integer>((0-256))

Specifies the number of buffer sectors to be used for system log input during rollback processing. When `0` is specified in this operand, HiRDB determines the number of rollback log input buffer sectors.

Specification guidelines

- We recommend that you specify `0` in this operand, in which case HiRDB will calculate an appropriate value automatically.
- If the specified value is too small, concurrent execution of rollbacks might be slowed. If the specified value is too large, the unit controller might use more shared memory than is necessary.

Tuning the specified value

Specify `0` in this operand. If specifying `0` leads to memory shortages, do not specify this operand.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is the appropriate value shown below:

- Standby-less system switchover (1:1) facility is used in the unit
  Number of alternate back-end servers × 2
- Standby-less system switchover (effects distributed) facility is used in the unit
  (*Number of host back-end servers* + *value of pd_ha_max_act_guest_servers*) × 2
- All other cases
  Number of servers in the unit × 2

Effects on individual estimation formulas

If the value of the `pd_log_rollback_buff_count` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formulas for shared memory used by a unit controller* under *Estimating the memory size required for a HiRDB/Parallel Server*

44) pd_log_auto_expand_size = *extension-amount-per-system-log-file-extension-trigger*[,*extension-limit*]
~<unsigned integer>((0-104857600))<<0,0>>(records)

Specify this operand when the system log file automatic extension facility is used.

Specifies the number of records to be added to the system log file for each extension trigger and an upper limit for extending file size.

For details about the system log file automatic extension facility, see the *HiRDB Version 9 System Operation Guide*.

Condition

`Y` must be specified in the `pd_large_file_use` operand or specification of this operand must be omitted.

Specification guidelines

- Specify in this operand the amount to be added per trigger based on the number of records specified when the system log file was created with the `-n` option of the `pdloginit` command. Determine 10 percent of the average number of records for all system log files and specify that value.
  ● In the case of a BES that uses the standby-less system switchover (1:1) facility, determine 10 percent of the average value including the system log files of the alternate BES and specify that value.
  ● In the case of a BES that uses the standby-less system switchover (effects distributed) facility, determine 10 percent of the average value including the system log files of the guest BES and specify that value.
- Normally, an extension limit is omitted.

Tuning the specified value

If the system log output volume exceeds the expanded size after automatic extension, the system log file could become full, resulting in a unitdown. In such a case, increase the specified value (number of records to be added per trigger). If extension processing requires so much time that it affects transaction performance, specify a smaller value.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed.

8.2.14 Operands related to synchronization point dump files

45) pd_spd_dual = Y | N

Specifies whether to use dual synchronization point dump files.

`Y`: Uses dual synchronization point dump files.

`N`: Does not use dual synchronization point dump files.

Advantage

When dual synchronization point dump files are used, HiRDB collects the same synchronization point dump in both dump files A and B. Even when an error occurs in one of the files when the collected synchronization point dump is being loaded, the synchronization point dump can be loaded from the other file, thus improving system reliability.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

To use dual synchronization point dump files, specify the name of synchronization point dump file B in the `pdlogadpf` operand.

46) pd_spd_assurance_msg = Y | N

Specifies whether the `KFPS02183-I` message is to be output when a synchronization point dump is completed.

`Y`: Output the message.

`N`: Do not output the message.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `Y`.

47) pd_spd_assurance_count = *number-of-guaranteed-valid-generations*
~<unsigned integer>((1-2))

Specifies the range of system log files to be saved during HiRDB operation as a protection against events such as a synchronization point dump file input error during system recovery. What is specified here is a number of synchronization point dump file generations; the specified value is referred to as the *number of guaranteed-valid generations*. The number of past generations of synchronization point dump files specified here are overwrite disabled.

Advantage

When `2` is specified as the number of guaranteed-valid generations and an error occurs in the most recent synchronization point dump file generation, the system can be recovered using the preceding synchronization point dump file generation, resulting in higher reliability.

Specification guidelines

- To improve reliability, specify `2` for the number of guaranteed valid generations. However, the number of overwrite disabled synchronization point dump files increases (to two).
- If reliability has been improved with the use of dual synchronization point dump files, we recommend that you omit this operand or specify `1` for it.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `1`.

Notes

- The minimum number of synchronization point dump files needed is *number-of-guaranteed-valid-generations* + 1.
- Specifying `2` increases the number of overwrite disabled synchronization point dump files. Moreover, the system log files that correspond to the overwrite disabled synchronization point dump files are also overwrite disabled. Consequently, specifying `2` for the number of guaranteed valid generations increases the number of overwrite disabled system log files. As a result, a shortage might occur in the number of system log files that can be swapped in. To prevent this, it might be necessary to re-evaluate the system log file capacity.

48) pd_spd_reduced_mode = *reduced-mode-operation-option*

~<unsigned integer>((0-2))

Specifies whether the reduced mode operation for synchronization point dump files is to be used.

Reduced mode operation is a facility for continuing processing if at least two synchronization point dump files can be used, even if an event such as a file error during HiRDB operation or restart reduces the number of synchronization point dump files to or below the number of required files (*number of guaranteed valid generations*# + 1).

#: Value specified for the `pd_spd_assurance_count` operand.

`0`: Do not use the reduced mode operation.

`1`: Use the reduced mode operation.

`2`: Use the reduced mode operation and issue a warning message whenever a synchronization point dump is being acquired during the reduced mode operation.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `0`.

49) pd_spd_reserved_file_auto_open = Y | N

Specifies whether a synchronization point dump file is to be opened automatically. When `Y` is specified and an error in a synchronization point dump file reduces the number of synchronization point dump files to the number of files necessary for operation (*number of guaranteed valid generations*# + 1), HiRDB opens a reserved file (if one is available), makes it overwritable, and continues processing. This is called *automatic opening of synchronization point dump file*.

#: Value specified for the `pd_spd_assurance_count` operand.

`Y`:

Open a synchronization point dump file automatically. A reserved file is opened when the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

`N`:

Do not open a synchronization point dump file. No reserved file is opened, even though the number of files falls below the number necessary for operation (*number of guaranteed valid generations* + 1).

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to other operands

This operand has a higher priority than the `pd_spd_reduced_mode` operand.

50) pd_spd_max_data_size = *synchronization-point-dump-file-buffer-size*

~<unsigned integer>((32000-4000000)) (bytes)

Specifies in bytes the size of the buffer (shared memory) to be used for synchronization point dump file input/output operations.

The value specified here affects the number of synchronization point dump file input/output operations.

Specification guidelines

- Normally, this operand need not be specified.
- When the value of this operand is increased, the number of synchronization point dump file input/output operations is reduced.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `32768`.

51) pd_log_sdinterval = *system-log-output-volume*[,interval]

Specifies the collection interval for synchronization point dumps. This operand can be specified based on the following information:

- Volume of system log information output since the previous synchronization point
- Amount of time that has elapsed since the previous synchronization point

***system-log-output-volume*: ~<unsigned integer>((100-100000)) (number of log blocks)**

Specifies an interval between synchronization point dumps in terms of the number of blocks of log information. A synchronization point dump is collected each time system log information equivalent to the number of log blocks specified here has been output.

***interval*: ~<unsigned integer>((0 or 10-1440)) (minutes)**

Specifies a synchronization point dump collection interval in terms of the number of minutes between synchronization point dumps.

- If `0` is specified for the interval, HiRDB does not use a time interval for collecting synchronization point dumps.
- If no transactions execute during an interval, no synchronization point dump is collected even though the amount of the time specified here has elapsed.

Specification guidelines

- This operand need not be specified if no specific amount of time is specified for restarting HiRDB.
- The value specified for this operand affects the amount of time required to restart HiRDB.
- Specifying a small value for this operand reduces the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps increases, online performance might deteriorate in some cases.

  Conversely, specifying a large value for this operand increases the amount of time required for database recovery during a HiRDB restart. However, because the frequency of synchronization point dumps decreases, online performance might improve in some cases.

Tuning the specified value

The synchronization point dump collection intervals can be checked with the statistics analysis utility; the relevant information is shown under `SYNC POINT GET INTERVAL` in the statistical information related to system operation. Use the average of the `SYNC POINT GET INTERVAL` values. If the synchronization point dump collection interval is determined to be too long, decrease the specification value; conversely, if it is determined to be too short, increase the specification value.

Operand default

If this operand is omitted, the specification for the same operand in the server common definition takes effect. If the same operand is also omitted in the server common definition, the following values are assumed:

- *system-log-output-volume*: `5000`
- *interval*: `60`

Note if `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand, `1000` is assumed as the default value for the system log output volume.

Note

- The synchronization point dump collection interval is determined on the basis of the volume of system log information that is output. Therefore, committing data from memory to a database takes a long time during intervals that have few updating transactions. If an error occurs during such a time, it will take longer to recover the transactions that were generated during that period. If this is a possibility, also use the `interval` value to set the synchronization point dump collection interval.
- When synchronization point dumps are collected, update pages are output from the global buffer, proportionately increasing the loads on the CPU, input/output processing, and lock time. For this reason, reducing the synchronization point dump collection interval might cause processing delays.

8.2.15 Operands related to server status files

52) pd_sts_file_name_1 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

:

pd_sts_file_name_7 = "*logical-file-name*", "*file-a-status-file-name*", "*file-b-status-file-name*"

Define server status files. Although the `pd_sts_file_name_2` to `7` operands can be omitted, the `pd_sts_file_name_1` operand cannot be omitted.

"*logical-file-name*" ~<identifier>((1-8 characters))

Specifies the logical file name of a status file for the back-end.

When a command that manipulates the status file is to be executed, the logical file name defined here must be specified.

"*file-a-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File A status file as an absolute path name.

"*file-b-status-file-name*" ~<path name>((up to 167 characters))

Specifies the name of the File B status file as an absolute path name.

Specification guidelines

- The files specified as File A and File B must be status files created with the `pdstsinit` command.
- If a file that has not been created with the `pdstsinit` command is specified, a virtual status file is created.
- If an error occurs in a status file, HiRDB swaps the status files. If no spare file is available for swapping, HiRDB (or a unit for a HiRDB/Parallel Server) terminates abnormally. For this reason, defining a large number of system files improves system reliability, but at the expense of increasing the amount of disk space that is required.
- The status files specified for File A and File B must have the same record length and capacity.

Operand rules

- Up to seven instances of this operand can be specified.
- A status file has a dual structure consisting of File A and File B; both must be specified.
- Environment variables cannot be used for the absolute path names of the File A and File B file names.
- The same names cannot be specified for the logical file name, the File A file name, and the File B file name.
- HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sts01`, `C:\hirdb\sysfile` is not case sensitive, but `sts01` is case sensitive.

Notes

- When HiRDB is started normally, the primary file (the file that was the primary file when HiRDB was terminated) is inherited. However, if there is no current file that can be inherited, for example because all status files have been initialized, the first status file that was specified among those specified by the `pd_sts_file_name_1` to `7` operands becomes the current file, and the remaining files become spare files if they can be opened. Any files that cannot be opened become reserved files.
- When HiRDB is restarted, the primary file (the file that was the primary file when HiRDB was terminated) is inherited.

Using a virtual status file

Specifying a virtual status file makes it possible to add a new status file while HiRDB is running.

For example, if the number of spare files is reduced because of errors in status files, virtual status files can be converted into spare files. The procedure for converting virtual status files into spare files follows.

Procedure

1. Use the `pdstsinit` command to create a status file in a HiRDB file system area for system files.
2. Use the `pdstsopen` command to open the status file.

These operations can be performed while HiRDB is running; there is no need to stop HiRDB.

- **Advantages and disadvantages**

  Defining a virtual status file reduces the amount of space required in the HiRDB file system area. However, a virtual status file cannot be added as a spare file if the HiRDB file system area for system files does not have sufficient space (sufficient space for adding the file), thus reducing system reliability.

  When virtual status files are not defined, the amount of space required in the HiRDB file system area is increased. However, because a swap destination is guaranteed when a file error occurs, system reliability is increased.

- **Notes**

When virtual status files are defined, HiRDB determines that a status file error has occurred when HiRDB is started. For this reason, HiRDB cannot start if `stop` (default value) is specified for the `pd_sts_initial_error` operand. When virtual status files are defined, specify `continue` or `excontinue` for the `pd_sts_initial_error` operand. It is also necessary before starting HiRDB to specify the current file in the `pd_sts_last_active_file` operand.

8.2.16 Operands related to server status files (when an error occurs)

For details about the measures to be taken when an error occurs in a status file, see the *HiRDB Version 9 System Operation Guide*.

53) pd_sts_initial_error = stop | continue | excontinue

When a server (single server, front-end server, dictionary server, or back-end server) starts, HiRDB performs a process of identifying the current server status file. This operand specifies the action that HiRDB takes when any of the following errors are detected during this identification process.

- No real server status file is found.
- An error is detected in the server status file.

The current file identification process is applied to the server status files specified by the `pd_sts_file_name_1` to `7` operands.

`stop`:

When an error is detected in a server status file during the current file identification process, the startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started. In this case, first take a corrective action for the status file in which the error was detected, and then start HiRDB.

`continue` or `excontinue`:

Even when an error is detected in the server status file during the current file identification process, the startup of the server is continued if the current file is normal. However, the startup might be stopped depending on the value specified for the `pd_sts_singleoperation` operand (whether operation continues with a single status file). The following table shows the relationship to the `pd_sts_singleoperation` operand.

- **Relationship to the pd_sts_singleoperation operand**

pd_sts_singleoperation operand value	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
`continue`	When an error is detected in a server status file, HiRDB cannot identify the current file, and thus startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	The HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.
`stop` (default value)	When an error is detected in a server status file, HiRDB identifies the current file and startup of the server is continued. However, if the primary and secondary files satisfy any of the conditions listed in the table below (cases in which HiRDB cannot identify the current file), startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	If HiRDB cannot identify the current file, the HiRDB administrator identifies the current file and specifies the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands. Afterwards, start HiRDB.

- **When HiRDB cannot identify the current file**

pd_sts_initial_error operand value	File A status	File B status
`continue`	Error shutdown	Error shutdown
	Error shutdown	Open (initial state)
	Error shutdown	No real file

pd_sts_initial_error operand value	File A status	File B status
	Open (initial state)	Error shutdown
	Open (initial state)	No real file
	No real file	Open (initial state)
	No real file	Error shutdown
	No real file	No real file
`excontinue`	Error shutdown	Error shutdown
	Error shutdown	No real file
	No real file	Error shutdown
	No real file	No real file

Therefore, to minimize actions by the HiRDB administrator (to reduce the number of cases in which both the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands must be specified), specify the following:

- `pd_sts_initial_error = excontinue`
- `pd_sts_singleoperation = stop`

Specification guidelines

The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_initial_error operand value	
	stop	continue or excontinue
Processing by HiRDB during server startup	When an error is detected in a server status file, startup of the server is stopped and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.	Even when an error is detected in a server status file, the startup of the server is continued if the current file is normal.
Specification guideline	To improve system reliability, specify `stop`.	To simplify the error-handling actions during HiRDB startup, specify `continue` or `excontinue`.
Advantage	Guarantees that all server status files of the server are normal when the server starts. Therefore, if an error occurs in the current file after HiRDB has started, it can be swapped to a spare file.	Even when an error is detected in a server status file during server startup, HiRDB can start with the remaining normal files only. Therefore, HiRDB stop time can be shortened. In this case, because the number of spare files has become small, it is necessary to immediately repair the status files containing errors.
Disadvantage	Possibility increases that an error in a server status file stops the startup of HiRDB.	Because HiRDB might be running with only a small number of spare files, system reliability is low. Depending on the number of spare files available, it might not be possible to swap server status files.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `stop`.

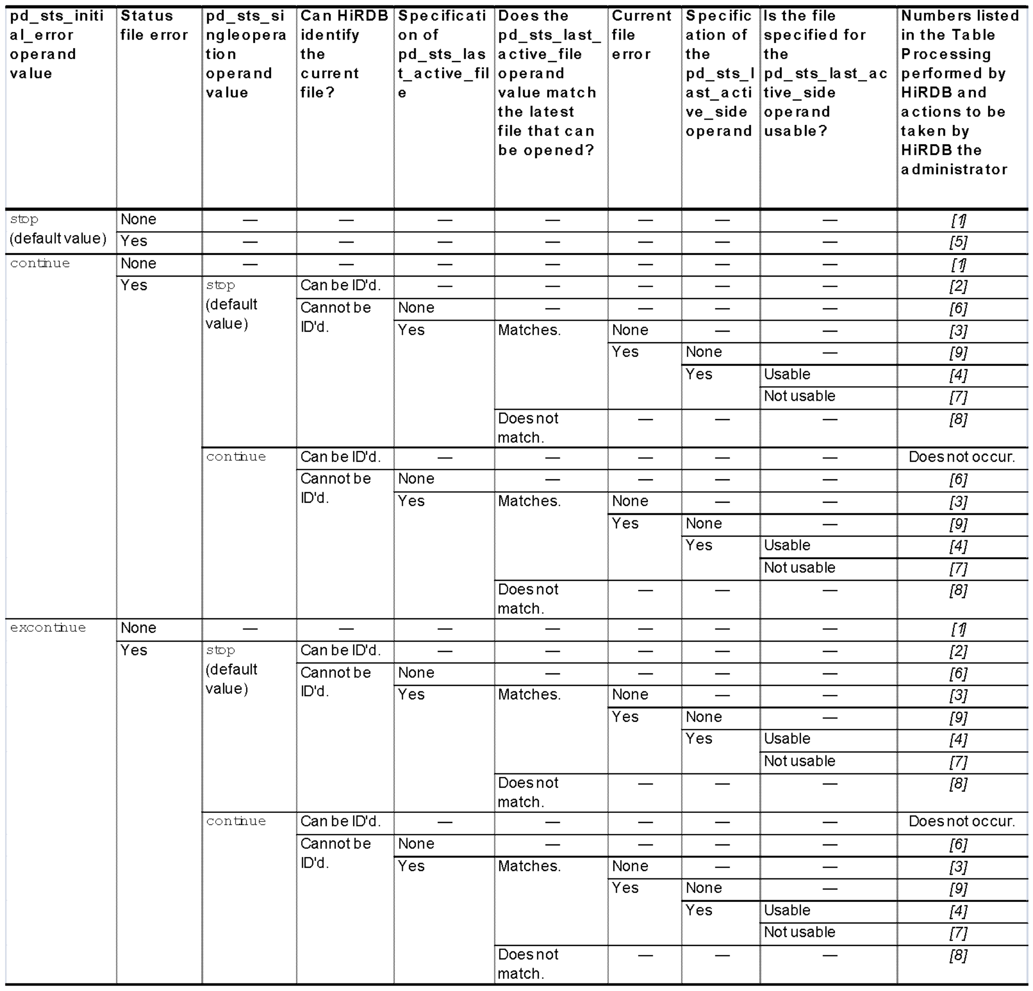
Notes

- If both current files are abnormal, startup of the server is stopped regardless of the value specified for this operand, and HiRDB (or the unit in the case of a HiRDB/Parallel Server) is not started.
- Before starting HiRDB, do not initialize the current file with the `pdstsinit` command. If the current file is initialized, HiRDB cannot be restarted.
- If `excontinue` is specified for this operand and all generations of the status files are in single-operation mode and open (initial status), HiRDB does not start.

Remarks

The following figure shows the values specified for this operand, the processing performed by HiRDB, and the actions to be taken by the HiRDB administrator.

- Values specified for this operand and the processing performed by HiRDB



pd_sts_initial_error operand value	Status file error	pd_sts_singleoperation operand value	Can HiRDB identify the current file?	Specification of pd_sts_last_active_file	Does the pd_sts_last_active_file operand value match the latest file that can be opened?	Current file error	Specification of the pd_sts_last_active_side operand	Is the file specified for the pd_sts_last_active_side operand usable?	Numbers listed in the Table Processing performed by HiRDB and actions to be taken by HiRDB the administrator
`stop` (default value)	None	—	—	—	—	—	—	—	[1]
	Yes	—	—	—	—	—	—	—	[5]
`continue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
`excontinue`	None	—	—	—	—	—	—	—	[1]
	Yes	`stop` (default value)	Can be ID'd.	—	—	—	—	—	[2]
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]
		`continue`	Can be ID'd.	—	—	—	—	—	Does not occur.
			Cannot be ID'd.	None	—	—	—	—	[6]
				Yes	Matches.	None	—	—	[3]
						Yes	None	—	[9]
							Yes	Usable	[4]
								Not usable	[7]
					Does not match.	—	—	—	[8]

Legend:

—: Not applicable (the condition does not affect the processing performed by HiRDB)

- Processing performed by HiRDB and the actions to be taken by the HiRDB administrator

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[1]	HiRDB startup processing continues.	None
[2]	HiRDB identifies the latest current file and continues the startup processing.	Make the file that is in the error-shutdown state into a spare file.
[3]	Using the file specified in the `pd_sts_last_active_file` operand as the current status file, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.
[4]	Using the files specified in the `pd_sts_last_active_file` and `pd_sts_last_active_side` operands as the current status files, HiRDB startup processing continues.	Make the file that is in the error-shutdown state into a spare file.

Corresponding No.	Processing performed by HiRDB	Action to be taken by the HiRDB administrator
[5]	Because `stop` is specified for the `pd_sts_initial_error` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000010` in message `KFPS01005-E`.
[6]	Because the current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000016` in message `KFPS01005-E`.
[7]	Because the normal current file identified by HiRDB does not match the file specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000017` in message `KFPS01005-E`.
[8]	Because the current file name identified by HiRDB does not match the file name specified in the `pd_sts_last_active_file` operand, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000015` in message `KFPS01005-E`.
[9]	Because the normal current file that was being used during the previous operation cannot be identified, HiRDB startup processing is stopped.	See the manual *HiRDB Version 9 Messages*, and take the corrective action indicated by reason code `0000000018` in message `KFPS01005-E`.

54) pd_sts_singleoperation = stop | continue

Specifies whether processing of server status files continues in the single-operation mode.

The status file single-operation mode means that processing is to continue using only the normal file (single file) when an error occurs in the status file and a spare file is not available. For details about the status file single-operation mode, see the *HiRDB Version 9 Installation and Design Guide*.

If an error occurs in one of the current files and a spare file is available, the status file is swapped and processing continues regardless of the value specified for this operand (operation in the single-operation mode does not occur).

`stop`:

Do not permit operation in the single-operation mode. If operation in the single-operation mode is necessary, HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated. If HiRDB is abnormally terminated, allocate spare files and then restart HiRDB.

`continue`:

Enable operation in the single-operation mode. When the single-operation mode goes into effect, the message `KFPS01044-I` is output. If an error occurs in the normal file during operation in the single-operation mode, or if HiRDB (or a unit for a HiRDB/Parallel Server) is abnormally terminated while the status file is being updated, HiRDB cannot be restarted. Therefore, when the single-operation mode goes into effect, immediately allocate spare files.

Specification guidelines

- We recommend that you specify `stop` to increase system reliability. Hitachi also recommends that you increase the number of spare files to guard against errors in the current files.
- The following table shows the specification guidelines, along with their advantages and disadvantages.

Item	pd_sts_singleoperation operand value	
	stop	continue
Specification guideline	To improve system reliability, specify `stop`.	Specify `continue` if it is important not to stop HiRDB.
Advantage	When an error occurs in one of the current files and a spare file is not available, operation in the single-operation mode does not occur and HiRDB is abnormally terminated. Consequently, the possibility of losing the content of the current files is reduced.	Even when an error occurs in one of the current files and a spare file is not available, processing can be continued. Therefore, the possibility that an error in the status file stops HiRDB is reduced.

Item	pd_sts_singleoperation operand value	
	stop	continue
Disadvantage	Possibility that an error in the status file stops HiRDB increases. However, increasing the number of spare files can reduce this possibility.	If an error occurs in the normal status file during operation in the single-operation mode or if HiRDB is abnormally terminated during updating of the status file, the content of the current file is lost and HiRDB cannot be restarted.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `stop`.

Relationship to other operands

The combination of the values specified for the `pd_sts_singleoperation` and `pd_sts_initial_error` operands determines the processing by HiRDB when an error occurs in the status file. Therefore, the values to be specified for these operands must be determined together.

55) pd_sts_last_active_file = "*logical-file-name*"

~<identifier> ((1-8 characters))

Specifies the name of the logical file to be used as the current status file at the time of the back-end server startup. HiRDB compares the file name specified in this operand with the file name selected by HiRDB to be the current file. If the file names match, HiRDB (or a unit for a HiRDB/Parallel Server) is started; otherwise, HiRDB (or a unit for a HiRDB/Parallel Server) is not started.

Conditions

The following conditions must be satisfied:

- `continue` or `excontinue` is specified in the `pd_sts_initial_error` operand.
- It cannot be determined if the current file selected by the HiRDB system was the most recent current file during the previous session.

Specification guidelines

1. To start HiRDB immediately after initializing all status files:
   From among the operable logical files specified in the `pd_sts_file_name1` to `7` operands, specify the one that has the smallest number. Forced startup will be used in this case, regardless of the previous termination mode.
2. When both of the current files are normal:
   Specify the name of the current file.# If HiRDB cannot be started even though the name of the current file is specified, the current file might have been initialized. In this case, first initialize all status files, and then use the method in step 1 above to start HiRDB (forced startup will be used, regardless of the previous termination mode).
3. When one of the current status files has an error:
   Use the method in step 2 above, with the following operands specified:
   - `pd_syssts_singleoperation = continue`
   - `pd_sts_last_active_side`
4. When both of the current status files have errors:
   Initialize all status files, then execute the method in step 1 above (forced startup will be used, regardless of the previous termination mode).
5. When a virtual status file is specified:
   Specify the name of the current file.#

#: The names of the current files (that were active at the end of the previous operation) can be determined from the following messages:

- `KFPS01001-I`
- `KFPS01010-E`
- `KFPS01011-I`
- `KFPS01063-I`

Of the status files displayed by these messages, the one that is reported in the message that was output most recently is the current file.

56) pd_sts_last_active_side = A | B

Specify this operand if you want to start the back-end server when one of the current files is in an error state. Specify the normal status file for this operand. HiRDB compares the file specified in this operand with the file selected by HiRDB. If the files match, HiRDB copies the contents of the normal status file to secondary File A and File B, then HiRDB switches the spare file in as the current file and starts the unit. If the files do not match, HiRDB (or a unit for a HiRDB/Parallel Server) is not started.

Conditions

The following operands must be specified:

- `pd_sts_initial_error = continue` or `excontinue`
- `pd_sts_last_active_file`

8.2.17 Operands related to the BES connection holding facility

For details about the BES connection holding facility, see the *HiRDB Version 9 System Operation Guide*.

57) pd_bes_connection_hold = Y | N

Specifies whether to use the BES connection holding facility.

`Y`: The BES connection holding facility is used. This setting can reduce the overhead of connection processing because the connection between the front-end and back-end servers is retained beyond synchronization points. Note that the number of memory segments, ports, and sockets that are also retained is the same as the number of connections that are retained.

`N`: The BES connection holding facility is not used. The connection between front-end and back-end servers is processed the first time an SQL statement is executed and the connection is released when a synchronization point is reached.

Specification guideline

For the specification guidelines, see *Application criteria* under *BES connection holding facility (HiRDB/ Parallel Server only)* in the *HiRDB Version 9 System Operation Guide*.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `N`.

Relationship to the client environment definition

The value of this operand can be changed for each client. To change the operand for a client, specify the `PDBESCONHOLD` operand in the client environment definition. For details about the `PDBESCONHOLD` operand, see the *HiRDB Version 9 UAP Development Guide*.

Note

When you use the BES connection holding facility, make sure that the following condition is satisfied:

number-of-processes-in-each-back-end-server (*value of the* `pd_max_bes_process` operand) ≥ *number-of-all-front-end-server-processes* (*value of the pd_max_users operand* × *number-of-front-end-servers*)

If this condition is not satisfied, a shortage in the number of back-end server processes might cause an SQL error. Furthermore, if you plan to execute a program such as a utility while HiRDB is running, make sure that you also include the number of back-end server processes required by the utility.

58) pd_bes_conn_hold_trn_interval = *back-end-server-connection-hold-time*
~<unsigned integer>((0-3600))(seconds)

Specifies the BES connection holding period in seconds.

When the BES connection holding facility is used, HiRDB monitors the period between the termination of a transaction and the execution of the next transaction. If this period is within the specified value, the connection between the front-end server and the back-end server is maintained. However, if this period exceeds the specified value, the connection between the front-end server and the back-end server is terminated after the transaction is terminated.

If `0` is specified for this operand, the period is not monitored. The connection between the front-end server and the back-end server is terminated only when the connection between the front-end server and a client is terminated by

`SQL DISCONNECT` (`xa_close` if the XA library is being used) or because the value of the `PDCWAITTIME` operand is exceeded.

Operand default

When this operand is omitted, the specification of the same operand in the server common definition is assumed. When the same operand is also omitted in the server common definition, the default is `1`.

8.2.18 Operands related to the work table files

59) pdwork -v "*HiRDB-file-system-area-name*"[,"*HiRDB-file-system-area-name*"]...

~<path name of up to 141 characters>

Specifies the names of HiRDB file system areas for work table files. Work table files are used for temporary storage of information during execution of SQL statements; they are created automatically by HiRDB. For details about the SQL statements that require work table files, see *Overview of the work table file* in the *HiRDB Version 9 Installation and Design Guide*.

You must not omit this operand, because doing so might prevent execution of SQL code that requires work table files.

Notes

- Specify in this operand the HiRDB file system area that was initialized using the `pdfmkfs` command.
- If the size of the work table file is large, specify a large HiRDB file system area. For details about how to estimate the work table file size, see the *HiRDB Version 9 Installation and Design Guide*.

  When an HiRDB file system area is set up initially with the `pdfmkfs -a` command, HiRDB will extend the area automatically whenever the amount of space specified in the `-n` option is used up. For details about the `pdfmkfs` command, see the manual *HiRDB Version 9 Command Reference*.
- The HiRDB file system areas for work table files must be different from the HiRDB file system areas for system files and RDAREAs.
- If more than one HiRDB file system area is specified in this operand, and one of these HiRDB file system areas generates an error when an attempt is made to create a work table file in it, that file system area will usually not be used thereafter. Instead, only the other specified HiRDB file system areas will be used.

  However, if subsequent attempts to create work table files fail in all of the other specified HiRDB file system areas (due to reasons such as insufficient space or an excessive numbers of files), creation of a work table file will be attempted in the HiRDB file system area that had been taken out of use. If a work table file can be created in it normally, that HiRDB file system area will then be used.

  Note that when a back-end server is shut down and restarted, a HiRDB file system area that was not being used because it had generated an error during work table file creation becomes usable again.

Operand rules

- At least one HiRDB file system area must be specified.
- A maximum of 16 HiRDB file system areas can be specified.
- This operand can be specified only once in the back-end server definition. If it is specified more than once, the first specification is effective.
- The specified HiRDB file system areas must not be used for other back-end servers.
- Do not specify a HiRDB file system area that is being used by a dictionary server.

8.2.19 Operands related to system log file configuration

60) pdlogadfg -d sys -g *file-group-name* [ONL]

Specifies a file group for a system log file. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign system log files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies a name for the file group. All file group names must be unique within a server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 200 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 200 times.

Notes

Before adding, modifying, or deleting this operand when HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while HiRDB Datareplicator is running, its extraction process might fail.

61) pdlogadpf -d sys -g *file-group-name* -a "*system-log-file-name*" [-b "*system-log-file-name*"]

Specifies the system log files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file that comprises the file group. Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

-b "*system-log-file-name*" ~<path name>((up to 167 characters))

Specifies an absolute path name as the name of the system log file B when dual system log files are to be used (`pd_log_dual = Y` specified). If `pd_log_dual = Y` is not specified, the system log file name is invalid, even if it is specified.

Specify the name of the system log file that was initialized using the `pdloginit` command. Note that the system log file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\log01`, `C:\hirdb\sysfile` is not case sensitive, but `log01` is case sensitive.

Notes

Before adding, modifying, or deleting this operand when HiRDB Datareplicator (extracted side) is linked, terminate the corresponding HiRDB Datareplicator. If this operand is modified while HiRDB Datareplicator is running, its extraction process might fail.

8.2.20 Operands related to synchronization point dump file configuration

62) pdlogadfg -d spd -g *file-group-name* [ONL]

Specifies a file group for synchronization point dump files. This operand cannot be omitted and must be specified. Use the `pdlogadpf` operand to assign synchronization point dump files to the file group specified here.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name* ~<<identifier>((1-8 characters))

Specifies a name for the file group. All file group names within a unit must be server.

`ONL`:

Specify this option to enable (open) this file group when HiRDB is running. You can specify 2 to 30 file groups with `ONL` specified.

Operand rule

This operand must be specified at least twice but no more than 60 times.

63) pdlogadpf -d spd -g *file-group-name* -a "*synchronization-point-dump-file-name*" [-b "*synchronization-point-dump-file-name*"]

Specifies the synchronization point dump files that comprise the file group. This operand cannot be omitted; it must be specified once for each file group.

Make sure that the `pdlogadfg` and `pdlogadpf` operands are specified in this order; otherwise, an error results.

-g *file-group-name*: ~<identifier>((1-8 characters))

Specifies the file group name specified by the `pdlogadfg` operand. The file group name must be unique within the unit.

-a "*synchronization-point-dump-file-name*": ~<path name> ((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file that comprises the file group. Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

-b "*synchronization-point-dump-file-name*": ~<path name> ((up to 167 characters))

Specifies an absolute path name as the name of the synchronization point dump file B when dual synchronization point dump files are to be used (`pd_spd_dual = Y` specified). If `pd_spd_dual = Y` is not specified, the synchronization point dump file name is invalid, even if it is specified.

Specify the name of the synchronization point dump file that was initialized using the `pdloginit` command. Note that the synchronization point dump file name must be unique within the unit.

Operand rule

HiRDB file system area names are not case sensitive, but HiRDB file names are case sensitive. For example, for `C:\hirdb\sysfile\sync01`, `C:\hirdb\sysfile` is not case sensitive, but `sync01` is case sensitive.

8.2.21 Operands related to Plug-ins

64) pdplgprm -n *plug-in-name* [-s *shared-memory-size*]

Specifies the name of a plug-in and the size of the memory to be shared by the plug-in. Omit this operand if no plug-ins are to be used or no plug-ins will run in back-end server.

Conditions

The plug-in specified here must have been registered in HiRDB with the `pdplgrgst` command.

-n *plug-in-name*: ~<identifier>((1-30 characters))

For details about the names of plug-ins that can be specified here, see the manuals for the plug-ins.

-s *shared-memory-size*: ~<unsigned integer> ((1-2000000)) <<0>> (kilobytes)

Specifies in kilobytes the size of the shared memory to be used by the plug-in. For details about the size of the shared memory to be used by the plug-in, see the manual for the applicable plug-in.

Effects on individual estimation formulas

If the value of the `pdplgprm` operand is changed, the following estimation formula is affected:

HiRDB Version 9 Installation and Design Guide:

- *Formula 1* under *Formulas for the size of the shared memory used by a back-end server*

9 UAP Environment Definition

This chapter explains the operands of the UAP environment definition.

9.1 Operand formats

A UAP environment definition defines the execution environment of a UAP. This section explains the formats used to specify the operands of a UAP environment definition. Note that the numbers in the following table correspond to the numbers assigned to the explanations of the individual operands in *9.2 Operand explanations*.

No.	Format	Modifiable after forced or abnormal termination?	Modifiable after planned termination?
1	`[set pd_uap_wait = Y	N]` (N underlined)	Y	Y

2	`[{{pdlbuffer -a` *local-buffer-name*
	`{-r` *RDAREA-name*`[,`*RDAREA-name*`]...`
	`| -i` *authorization-identifier*`.`*index-identifier*`}`
	`-n` *buffer-sector-count*
	`[-p` *batch-input-maximum-page-count*`]}}]`
		Y	Y

Y: The specification value can be modified.

9.2 Operand explanations

1) pd_uap_wait = Y | N

Specifies the action to be taken by the UAP when the RDAREA or index to be accessed using a local buffer is being used by another user.

`Y`: Waits for the other user to terminate processing. The UAP goes into a wait state.

`N`: If the RDAREA or index to be accessed using a local buffer is being used by another user, an error is returned to the UAP.

If this operand is specified together with two or more `pdlbuffer` operands, the specification of this operand is valid for all the local buffers specified.

Condition

The `pdlbuffer` operand must be specified.

2) `pdlbuffer -a` *local-buffer-name*

{`-r` *RDAREA-name*[,*RDAREA-name*]...| `-i` *authorization-identifier.index-identifier*}

`-n` *buffer-sector-count*

[`-p` *batch-input-maximum-page-count*]

Specifies the local buffer used by a UAP. A local buffer is an area used for inputting and outputting table or index data located in a disk, and is allocated in the process private memory of a single server or back-end server. The maximum number of local buffers you can define is 100. For a HiRDB/Parallel Server, you can define up to 100 local buffers for each back-end server.

When a local buffer is specified, various types of resources are locked according to the specification of the `pdlbuffer` operand as shown as follows.

pdlbuffer operand specification	Locked resource	Lock mode
The `-r` option of the `pdlbuffer` operand is specified.	Specified RDAREA	`EX`
The `-i` option of the `pdlbuffer` operand is specified.	RDAREA for index	`SU`
	Table for index	`EX`
	Specified index	`EX`

Condition

A global buffer must first be allocated to the RDAREA or index to be specified as a local buffer, using the `pdbuffer` operand.

Application criterion

Define a local buffer when all of the following conditions are satisfied:

- A large volume of data is searched or updated.
- The access target RDAREA is not accessed by other UAPs.

Because UAPs that are always connected to HiRDB have a major impact on the system (memory shortage or process lock), do not define local buffers for them.

Note

Note the following for a HiRDB/Parallel Server:

- If a server process is abnormally terminated when a local buffer is being used, the unit might terminate abnormally, and the abort code `Phb3008` might be output.
- If a page is being updated when the server process is abnormally terminated, recovery using a rollback process might not be possible. In this case, recovery is performed when the unit is restarted.
- For the processing by HiRDB and the action to be taken when an error occurs when a local buffer is being used, see the *HiRDB Version 9 System Operation Guide*.

`-a` *local-buffer-name*

~<*identifier*>((1-16 characters))

Specifies a local buffer name. The name specified here must be unique. However, the same name can be specified for local buffers that are in different UAP environment definition files.

`-r` ***RDAREA-name*[,*RDAREA-name*]...**

~<*identifier*>((1-30 characters))

Specifies the name of the RDAREA to which a local buffer for data is allocated. Only user RDAREA names can be specified.

The name specified here must be unique. However, the same name can be specified for RDAREAs that are in different UAP environment definition files.

Specification guidelines

When multiple RDAREAs are allocated to a single local buffer, the maximum page size of the RDAREAs becomes the buffer size. Therefore, allocating RDAREAs having the same or approximately the same size to the local buffer can reduce the number of inputs and outputs. However, with the types of RDAREAs described as follows, allocating them to different local buffers can reduce the number of inputs and outputs, even if these RDAREAs have the same size.

- RDAREAs that store tables for different purposes
- RDAREAs that are accessed randomly or sequentially most of the time

Operand rules

- If the RDAREA name includes a character that is not alphanumeric, enclose it in quotation marks.
- The maximum number of RDAREAs you can define for a single local buffer is 4,096.

`-i` ***authorization-identifier.index-identifier***

***authorization-identifier*:~<*identifier*>((1-8 characters))**

***index-identifier*:~<*identifier*>((1-30 characters))**

Specifies the index name (*authorization-identifier.index-identifier*) to which the local buffer for the index is allocated.

The name specified here must be unique. However, the same name can be specified for indexes that are in different UAP environment definition files.

Operand rule

If the authorization identifier or index identifier includes a character that is not alphanumeric, enclose it in quotation marks.

Specification guidelines

Specify an index that is frequently used. Allocating a local buffer to a frequently used index increases the degree of memory residency for index pages, and thus can reduce the number of inputs and outputs.

In particular, allocating an index defined in a cluster key or unique key to a local buffer can have a major positive effect. Note that because the index identifier of a cluster key is determined by HiRDB, check the index identifier by searching the dictionary table (`INDEX_NAME` column of the `SQL_INDEXES` table) after a table has been defined.

Notes

A local buffer for an index that specifies a temporary table index name is invalid.

`-n` ***buffer-sector-count***

~<unsigned integer>((4-125000))

Specifies the number of local buffer sectors.

Specification guidelines

- A local buffer is allocated in the process private memory. Therefore, allocating an unnecessarily large area to the local buffer causes frequent paging when the rest of the memory is used, resulting in poor performance.
- Note that specifying too large a buffer sector count might make it impossible to allocate process private memory.
- For a sequentially-accessing UAP, no buffering effect can be expected. Therefore, specify the minimum value of 4 for the buffer sector count.

`-p` ***batch-input-maximum-page-count***

~<unsigned integer>((2-256))

Specifies the maximum number of pages that can be input by the prefetch facility in a single batch. If this operand is omitted, the prefetch facility does not work.

Specification guidelines

By taking the cost-performance tradeoff between memory size and input/out time reduction effect into consideration, specify a value that satisfies the following condition:

$a \times b$ = 64 - 128 (kilobytes)

a: Size of the RDAREA that stores the data or index to be prefetched

b: Maximum number of pages that can be input in a single batch

If the segment size of the RDAREA is 1, specifying this option has no effect. Therefore, do not specify it.

Appendixes

A. Operand Specification Values

This appendix provides an overview of what is specified in each of the HiRDB system definition operands. Note that the operands for the following definitions are not described in this appendix.

- Foreign server information definition (HiRDB External Data Access facility)
- Hub optimization information definition (HiRDB External Data Access facility)

(1) Operands related to system configuration

Operand explanation	Operand name
Specifies a HiRDB identifier.	`pd_system_id`
Specifies a HiRDB port number.	`pd_name_port`
Specifies the first HiRDB file in the master directory RDAREA.	`pd_master_file_name`
Allocates a unit to a host.	`pdunit`
Allocates a server to a host.	`pdstart`
Specifies a unit identifier.	`pd_unit_id`
Specifies a standard host name.	`pd_hostname`

(2) Operands related to maximum concurrent executions

Operand explanation	Operand name
Specifies the maximum number of concurrent connections.	`pd_max_users`
Specifies the maximum number of server processes that can be activated concurrently within a single unit.	`pd_max_server_process`
Specifies the maximum number of tables that can be accessed concurrently by a transaction.	`pd_max_access_tables`
Modifies the maximum number of utilities that can be executed concurrently.	`pd_utl_exec_mode`
Specifies the maximum number of `pdreclaim` commands (with the `-p` option specified) that can be executed concurrently.	`pd_max_commit_write_reclaim_no`

(3) Operands related to processes

Operand explanation	Operand name
Specifies the maximum number of processes that can be activated in any back-end server.	`pd_max_bes_process`
Specifies the maximum number of processes that can be activated in any dictionary server.	`pd_max_dic_process`
Specifies the number of resident processes that can be activated at server startup.	`pd_process_count`
Specifies in minutes the interval checking for nonresident server processes in HiRDB that are to be stopped. This facility is applied when the number of executing server processes exceeds the number of processes that can be made resident (value specified by the `pd_process_count` operand).	`pd_server_cleanup_interval`
Specifies the number of processes necessary for asynchronous READ operations when using the asynchronous READ facility. For a HiRDB/Parallel Server, this	`pd_max_ard_process`

Operand explanation	Operand name
operand specifies the number of processes per server (back-end server or dictionary server).	
Specifies the number of processes to be processed in parallel when the facility for parallel writes in deferred write processing is used.	`pd_dfw_awt_process`

(4) Operands related to work tables

Operand explanation	Operand name
Specifies the method of allocating buffers for creating work tables for temporary storage of information during SQL execution.	`pd_work_buff_mode`
Specifies the size of the work table buffer to be used for SQL execution.	`pd_work_buff_size`
Specifies the upper limit for expanding the work table buffer when the space in this buffer becomes insufficient.	`pd_work_buff_expand_limit`
Specifies the names of HiRDB file system areas for work table files.	`pdwork`

(5) Operands related to HiRDB startup

Operand explanation	Operand name
Specifies the HiRDB system startup method.	`pd_mode_conf`
Specifies the amount of time to wait for completion of the `pdstart` command execution.	`pd_system_complete_wait_time`
Specifies the maximum amount of time to wait for completion of HiRDB start preparation.	`pd_start_time_out`

(6) Operands related to reduced activation

Operand explanation	Operand name
Specifies whether reduced activation is to be used.	`pd_start_level`
Specifies the amount of time to wait for receipt of a notice of reduced activation startup.	`pd_reduced_check_time`
Specifies the names of units that need not be started because of errors or other problems during HiRDB startup.	`pd_start_skip_unit`

(7) Operands related to HiRDB processing

Operand explanation	Operand name
Specifies the timing for committing database changes to a file.	`pd_dbsync_point`
Specifies the timing for committing to file updates in the following types of RDAREAs: • Master directory RDAREA • Data directory RDAREA • Data dictionary RDAREAs • Data dictionary LOB RDAREAs • Registry RDAREAs	`pd_system_dbsync_point`

Operand explanation	Operand name
• Registry LOB RDAREAs	
Specifies whether the following write processing is to be skipped: Writing of the update buffer's contents into the database when a request to reference an update buffer is issued during synchronization point acquisition processing (if the write operation is not skipped, the writing is handled by the server process that executed the transaction that issued the referencing request).	`pd_dbsync_altwrite_skip`
Specifies whether, when HiRDB is started, to activate the post-processing process, which executes post-processing when a HiRDB process terminates abnormally.	`pd_process_terminator`
Specifies the number of post-processing processes to be started when starting HiRDB.	`pd_process_terminator_max`
Specifies the size of the desktop heap used per process in HiRDB.	`pd_process_desktopheap_size`
If contention occurs in the HiRDB server process during concurrent execution of UAPs, processing requests might sometimes be temporarily queued. This operand specifies what HiRDB must do in this case.	`pd_server_entry_queue`
Specifies the type of sleep function (for waiting for a specified amount of time) to be used when acquiring a lock between threads.	`pd_thdlock_sleep_func`
Specifies a thread lock release notification method.	`pd_thdlock_wakeup_lock`
Specifies in microseconds the interval at which to check for thread lock release.	`pd_thdlock_pipe_retry_interval`
Specifies in microseconds the thread lock sleep time.	`pd_thdlock_retry_time`
Specifies a spin count for thread spin lock.	`pd_thdspnlk_spn_count`
Specifies the page access mode to be used for database retrieval.	`pd_pageaccess_mode`
Specifies the method to be used for checking for RDAREA hold.	`pd_cmdhold_precheck`
Specifies the processing to be performed by HiRDB when an input/output error occurs in an RDAREA (excluding the master directory RDAREA).	`pd_db_io_error_action`
Specifies whether to hide the error cause in the message that is output when a connection attempt fails.	`pd_connect_errmsg_hide`
Specifies whether a loopback address is to be used for `bind()` when the receiving port is generated.	`pd_rpc_bind_loopback_address`
Specifies whether the error messages output when a server process is forcibly terminated are to be changed.	`pd_cancel_down_msgchange`

(8) Operands related to full recovery processing

Operand explanation	Operand name
Specifies the number of processes to be recovered (`REDO` processes) during full recovery processing. For a HiRDB/Parallel Server, this operand specifies the number of processes to be recovered (`REDO` processes) per server (dictionary server or back-end server).	`pd_max_recover_process`
Specifies whether to output to a database the pages updated after a synchronization point during full recovery processing.	`pd_redo_allpage_put`

(9) Operands related to transaction decision processing

Operand explanation	Operand name
Specifies whether an undecided transaction that has branched from a transaction is to be decided automatically when a unit is restarted. An undecided transaction occurs when a unit terminates abnormally while the latter transaction's first phase of `COMMIT` is incomplete.	`pd_trn_rerun_branch_auto_decide`
Specifies the interval (in seconds) for sending an automatic decision instruction to a branched transaction when the previous send operation has failed for some reason.	`pd_trn_send_decision_intval_sec`
Specifies the interval (in minutes) for sending an automatic decision instruction to a branched transaction when the previous send operation has failed for some reason.	`pd_trn_send_decision_interval`
Specifies the maximum amount of time to wait for a decision completion notice to be returned after an automatic decision instruction has been sent to a branched transaction.	`pd_trn_send_decision_retry_time`
Specifies the maximum amount of time to wait for receiving communication (prepare, commit instruction, or response) between transaction branches during transaction synchronization point processing executed in the HiRDB server process.	`pd_trn_watch_time`
Specifies the transaction recovery message queue size.	`pd_trn_rcvmsg_store_buflen`
Specifies whether to use one-phase commit in a HiRDB/Parallel Server's commitment control.	`pd_trn_commit_optimize`
Specifies the interval at which the `KFPS02235-I` message, indicating that transaction rollback processing is in progress, is to be output.	`pd_trn_rollback_msg_interval`
Specifies the maximum amount of time for the front-end server to wait to receive a rollback completion response from a transaction branch. This operand also specifies the period of time during which the resending of rollback instruction is to be repeated in the event of a communication error.	`pd_trn_rollback_watch_time`

(10) Operands related to the SQL specifications

Operand explanation	Operand name
Specifies whether errors are to be suppressed during operation.	`pd_overflow_suppress`
Specifies a space conversion level when the space configuration facility is used.	`pd_space_level`
Specifies whether the facility for conversion to a `DECIMAL` signed normalized number is to be used.	`pd_dec_sign_normalize`
Specifies the maximum precision for the operation result in the case of a `DECIMAL` type operation result of no more than 29 digits.	`pd_sql_dec_op_maxprec`
Specifies whether to allow the set operator `MINUS` in SQL statements.	`pd_sql_mode`
Specifies whether to allow simple comments in SQL statements.	`pd_sql_simple_comment_use`

(11) Operands related to SQL optimization

Operand explanation	Operand name
Specifies SQL optimization options.	`pd_optimize_level`
Specifies SQL extension optimizing options.	`pd_additional_optimize_level`

Operand explanation	Operand name
Specifies the hashing method to be used when `hash join, subquery hash execution` is specified for the SQL extension optimizing option.	`pd_hashjoin_hashing_mode`
Specifies the size of the hash table to be used when *application of hash-execution of a hash join or a subquery* is specified as an SQL optimization option.	`pd_hash_table_size`
Specifies one of the following HiRDB processing methods to be used for executing an SQL statement that uses a work table: • Lock acquisition method when AND multiple indexes are used • Suppression of message output during automatic extension of the work table buffer	`pd_work_table_option`
Specifies the maximum number of combinations of narrowing values that can be applied to an `ATS` or `RANGES` search condition when an `IN` predicate or a quantified predicate (= `ANY` (table subquery) or = `SOME` (table subquery)) is specified in an indexed search.	`pd_apply_search_ats_num`
Specifies back-end servers that can be used as floating servers.	`pd_floatable_bes`
Specifies back-end servers that are not to be used as floating servers.	`pd_non_floatable_bes`

(12) Operands related to the facility for output of extended SQL error information

Operand explanation	Operand name
Specifies whether to use the facility for output of extended SQL error information.	`pd_uap_exerror_log_use`
Specifies the directory in which to store SQL error report files.	`pd_uap_exerror_log_dir`
Specifies the maximum size of an SQL error report file.	`pd_uap_exerror_log_size`
Specifies the maximum data size for the parameter information to be output to a client error log file or an SQL error report file.	`pd_uap_exerror_log_param_size`

(13) Operands related to the SQL reserved word deletion facility

Operand explanation	Operand name
Specifies the name of an SQL reserved word deletion file when the SQL reserved word deletion facility is used.	`pd_delete_reserved_word_file`

(14) Operands related to command execution from SQL

Operand explanation	Operand name
Specifies the authorization identifiers that can use the SQL `CALL COMMAND` statement to execute commands and utilities.	`pd_sql_command_exec_users`

(15) Operands related to detailed SQLSTATE values

Operand explanation	Operand name
Specifies whether to output detailed `SQLSTATE` values.	`pd_standard_sqlstate`

(16) Operands related to narrowed retrieval

Operand explanation	Operand name
Specifies the maximum number of users who can own lists concurrently.	`pd_max_list_users`
Specifies the maximum number of lists that one user can create.	`pd_max_list_count`
Specifies the list initialization (deletion) timing. Normally, lists are initialized when HiRDB is started (including restart). You can use this operand to change the initialization timing.	`pd_list_initialize_timing`

(17) Operands related to system monitoring

Operand explanation	Operand name
Specifies the monitoring time (in minutes) for monitoring the execution time of the following utilities: • Database load utility (`pdload` command) • Database reorganization utility (`pdrorg` command) If the execution of a utility is not terminated within the monitoring time specified by this operand, the executing utility is forcibly terminated, and the error information for identifying the cause of no response is output.	`pd_utl_exec_time`
Specifies a maximum execution time for SQLs that are executed in a HiRDB server process. If execution of an SQL statement is not completed within the specified amount of time, execution of that SQL statement is terminated.	`pd_watch_time`
If messages cannot be extracted from the message queue within the time specified by this operand, a warning message (`KFPS00888-W` or `KFPS00889-E`) is output (the message queue monitoring facility).	`pd_queue_watch_time`
Specifies the processing to be performed by HiRDB when messages cannot be extracted from the message queue within the time specified by the `pd_queue_watch_time` operand.	`pd_queue_watch_timeover_action`
If abnormal terminations of server processes exceed the value specified in this operand, HiRDB (applicable unit for a HiRDB/Parallel Server) terminates abnormally (the facility for monitoring abnormal process terminations).	`pd_down_watch_proc`
Specifies the interval at which the other host's operation status is to be checked.	`pd_host_watch_interval`
Specifies the maximum amount of time to wait for the next request from a HiRDB client after the HiRDB server returns a response to a request from a Windows-compatible HiRDB client.	`pd_watch_pc_client_time`
Specifies the number of successive synchronization point dumps that can be skipped (number of skips in a single transaction).	`pd_spd_syncpoint_skip_limit`
Specifies the maximum number of times synchronization point dumps can be skipped (number of skips per transaction) when synchronization point dump acquisition is delayed by deferred write processing.	`pd_dfw_syncpoint_skip_limit`
Specifies whether a warning message related to resource usage is to be issued.	`pd_watch_resource`
Specifies whether to output the warning message when the number of concurrent connections reaches or exceeds the specified percentage of the maximum value specified by the `pd_max_users` operand.	`pd_max_users_wrn_pnt`
Specifies whether a warning message related to the number of concurrently accessible base tables specified by the `pd_max_access_tables` operand is to be issued.	`pd_max_access_tables_wrn_pnt`
Specifies whether to output the warning message when the number of RDAREAs reaches or exceeds the specified percentage of the maximum value specified by the `pd_max_rdarea_no` operand.	`pd_max_rdarea_no_wrn_pnt`

Operand explanation	Operand name
Specifies whether to output the warning message when the number of HiRDB files comprising an RDAREA reaches or exceeds the specified percentage of the maximum value specified by the `pd_max_file_no` operand.	`pd_max_file_no_wrn_pnt`
Specifies whether a warning message related to the HiRDB file system areas for work table files specified by the `pdwork` operand is to be issued.	`pdwork_wrn_pnt`
Specifies as a percentage of the number of users who can create lists (as specified by the `pd_max_list_users` operand) the point at which the number of users actually using lists is to cause a warning message to be issued.	`pd_max_list_users_wrn_pnt`
Specifies as a percentage of the number of lists that can be created per user (as specified by the `pd_max_list_count` operand) the point at which the number of lists created by a user is to cause a warning message to be issued.	`pd_max_list_count_wrn_pnt`
Specifies as a percentage of the number of lists that can be created at a server the point at which a warning message is to be issued.	`pd_rdarea_list_no_wrn_pnt`

(18) Operands related to lock

Operand explanation	Operand name
Specifies whether deadlock information is to be output.	`pd_lck_deadlock_info`
Specifies the maximum amount of time for monitoring lock wait time.	`pd_lck_wait_timeout`
Specifies the method by which HiRDB is to detect lock release.	`pd_lck_release_detect`
Specifies the interval at which HiRDB is to reference the lock management area.	`pd_lck_release_detect_interval`
Specifies the processing method to be used when `WITHOUT LOCK NOWAIT` search is executed.	`pd_nowait_scan_option`
Specifies the number of users who must be waiting for lock release in order for a warning message to be issued.	`pd_lck_queue_limit`
Specifies whether deadlock priorities are to be used.	`pd_deadlock_priority_use`
Specifies the deadlock priority value of a command.	`pd_command_deadlock_priority`
Specifies the method for creating a locked resource for an index key value.	`pd_key_resource_type`
Specifies the amount of lock pool space to be used by a server.	`pd_lck_pool_size`
Specifies the amount of lock pool to be used by a front-end server.	`pd_fes_lck_pool_size`
Specifies the number of lock pool partitions that are to be used per server for distributing lock processing.	`pd_lck_pool_partition`
Specifies the number of lock pool partitions that are to be used per front-end server for distributing lock processing.	`pd_fes_lck_pool_partition`
Specifies the number of resources to be locked for the tables and RDAREAs that are to be held across transactions.	`pd_lck_until_disconnect_cnt`
Specifies the maximum number of holdable cursors that can be concurrently open for each transaction when holdable cursors are used for a table for which a `LOCK` statement with `UNTIL DISCONNECT` specification is not executed.	`pd_max_open_holdable_cursors`
Specifies the number of hash table entries to be used in the lock pool.	`pd_lck_hash_entry`
Specifies the locking method for a B-tree index (whether *index key value no lock* is to be executed).	`pd_indexlock_mode`
Specifies the lock release wait control mode for uncommitted delete data.	`pd_lock_uncommited_delete_data`
Specifies whether row identifiers are to be reused when row data is inserted.	`pd_dbreuse_remaining_entries`

Operand explanation	Operand name
Specifies whether to check for deadlocks.	`pd_lck_deadlock_check`
Specifies the interval between checks for deadlocks.	`pd_lck_deadlock_check_interval`
Specifies an interval for unlocking global buffers when global buffer locking occurs during synchronization point processing.	`pd_dbsync_lck_release_count`

(19) Operands related to buffers

Operand explanation	Operand name
Specifies the size of the buffer area in which SQL objects are to be placed.	`pd_sql_object_cache_size`
Specifies whether the control areas for the buffers for table definition information, buffers for view analysis information, buffers for user-defined type information, and buffers for routine definition information are to be allocated in a single batch at the time of HiRDB activation or in a single batch at the time of transaction activation.	`pd_def_buf_control_area_assign`
Specifies the maximum stack size to be used by one thread.	`pd_thread_max_stack_size`
Specifies the stack extension size per thread.	`pd_thread_stack_expand_size`
Specifies the size of the buffer for table definition information.	`pd_table_def_cache_size`
Specifies the size of the buffer for user privilege information.	`pd_auth_cache_size`
Specifies the size of the buffer for view analysis information.	`pd_view_def_cache_size`
Specifies the size of the buffer for user-defined type information.	`pd_type_def_cache_size`
Specifies the size of the buffer for routine definition information.	`pd_routine_def_cache_size`
Specifies the size of a buffer to be used to store registry information.	`pd_registry_cache_size`

(20) Operands related to shared memory

Operand explanation	Operand name
Specifies whether the shared memory to be used by the HiRDB is to be fixed in the memory.	`pd_shmpool_attribute`
Specifies the control method for shared memory used by HiRDB.	`pd_shmpool_control`
Specifies whether the shared memory to be used by the global buffer is to be fixed in the real memory.	`pd_dbbuff_attribute`
Specifies whether shared memory-related report messages are to be issued when HiRDB starts.	`pd_shared_memory_report`
Specifies the maximum number of segments to be allocated per server if the shared memory for global buffers is allocated based on the segment size specified in the `SHMMAX` operand when HiRDB starts.	`pd_max_dbbuff_shm_no`
Specifies the size of the shared memory to be used by a single server.	`pd_sds_shmpool_size`
Specifies the size of the shared memory to be used by the dictionary server.	`pd_dic_shmpool_size`
Specifies the size of the shared memory to be used by a back-end server.	`pd_bes_shmpool_size`
Specifies the upper limit of the segment size for the shared memory for the global buffer pool.	`SHMMAX`

(21) Operands related to RDAREAs

Operand explanation	Operand name
Specifies the maximum number of RDAREAs allowed.	`pd_max_rdarea_no`
Specifies the maximum number of HiRDB files that comprise an RDAREA.	`pd_max_file_no`
Specifies triggers for issuing a warning message (`KFPH00211-I` or `KFPA12300-I`) about RDAREA segment usage or about area usage by a HiRDB file that is being extended (`KFPH22037-W`).	`pd_rdarea_warning_point`
Specifies whether to issue an RDAREA segment usage warning message (`KFPH00211-I` or `KFPA12300-I`).	`pd_rdarea_warning_point_msgout`
Specifies when to automatically extend an RDAREA (the trigger) when the RDAREA auto-extension facility is being used.	`pd_rdarea_extension_timing`
Specifies whether to initialize the extended area when an RDAREA is extended automatically.	`pd_rdarea_expand_format`
Specifies whether to use the `DEFER` or `SCHEDULE` attribute as the RDAREA opening trigger.	`pd_rdarea_open_attribute` `_use`
Specifies the standard value for the RDAREA opening trigger attribute.	`pd_rdarea_open_attribute`
Specifies whether to use a shared RDAREA.	`pd_shared_rdarea_use`
Specifies the action when a file access error occurs while accessing an RDAREA.	`pd_db_access_error_action`
Specifies the action to be taken when a physical error (input/output error or file open error) occurs while accessing an RDAREA.	`pd_db_hold_action`

(22) Operands related to global buffers

Operand explanation	Operand name
Specifies the LRU management method for the global buffer.	`pd_dbbuff_lru_option`
Specifies whether to use LRU management for global buffers when executing a UAP that accesses a column for which `BINARY` type was specified.	`pd_dbbuff_binary_data_lru`
Specifies whether to dynamically modify the global buffer.	`pd_dbbuff_modify`
Specifies the method of detecting lock release of the global buffer.	`pd_dbbuff_lock_release_detect`
Specifies the number of spins during lock acquisition wait processing for the global buffer.	`pd_dbbuff_lock_spn_count`
Specifies the interval during lock acquisition wait processing for the global buffer.	`pd_dbbuff_lock_interval`
Specifies the interval at which to check the global buffer occupation state.	`pd_dbbuff_wait_interval`
Specifies the maximum number of spin loops for global buffer occupation state checking.	`pd_dbbuff_wait_spn_count`
Specifies the deferred write trigger event as a percentage of the updated buffers.	`pd_dbbuff_rate_updpage`
Specifies the acquisition level for database buffer control information traces.	`pd_dbbuff_trace_level`
Defines the RDAREAs to be allocated to a global buffer.	`pdbuffer`
Specifies the maximum number of global buffers for dynamic addition.	`pd_max_add_dbbuff_no`
Specifies the maximum number of shared memory segments for dynamic addition.	`pd_max_add_dbbuff_shm_no`

(23) Operands related to in-memory data processing

Operand explanation	Operand name
Specifies the maximum number of in-memory RDAREAs.	`pd_max_resident_rdarea_no`
Specifies the maximum number of shared memory segments used by an in-memory data buffer.	`pd_max_resident_rdarea_shm_no`

(24) Operands related to table or index reservation count

Operand explanation	Operand name
Specifies the number of tables that can use the free space reusage facility.	`pd_assurance_table_no`
Specifies the number of indexes that can be used.	`pd_assurance_index_no`

(25) Operands related to referential and check constraints

Operand explanation	Operand name
Specifies whether to specify a constraint name definition before or after constraint definition in a referential constraint or check constraint.	`pd_constraint_name`
Specified when a referential constraint or check constraint is used.	`pd_check_pending`

(26) Operands related to temporary tables

Operand explanation	Operand name
Specifies the maximum number of temporary table RDAREAs.	`pd_max_tmp_table_rdarea_no`
Specifies the maximum number of temporary tables and temporary table indexes that can be used at any one time for each server.	`pd_max_temporary_object_no`
Specifies to change when temporary table RDAREAs are initialized to a time other than when HiRDB starts.	`pd_tmp_table_initialize_timing`

(27) Operands related to HiRDB file system areas

Operand explanation	Operand name
Specifies whether to use a HiRDB file system area with a size of 2,048 MB or greater.	`pd_large_file_use`
Specifies whether to use the no-cache access method to access HiRDB file system areas.	`pd_ntfs_cache_disable`

(28) Operands related to system log files

Operand explanation	Operand name
Specifies whether dual system log files are to be used.	`pd_log_dual`
Specifies the processing to be performed by HiRDB when the available space in the system log file falls below the warning level (the facility for monitoring the free area for the system log file).	`pd_log_remain_space_check`

Operand explanation	Operand name
Specifies as absolute path names the unload file output directories or HiRDB file system areas when the automatic log unloading facility is to be used for the system log.	`pd_log_auto_unload_path`
Specifies whether the automatic log unloading facility is to be restarted when the system log files are swapped following the issuance of s message indicating that the automatic log unloading facility stopped due to an error (message `KFPS01150-E`).	`pd_log_auto_unload_restart`
Specifies whether single operation of the system log files is to be used.	`pd_log_singleoperation`
Specifies whether a reserved file is to be used when none of the opened files can be used for swapping.	`pd_log_rerun_reserved_file_open`
Specifies whether system log files are to be swapped during a restart.	`pd_log_rerun_swap`
Specifies a wait time for swapping of system log files to be completed.	`pd_log_swap_timeout`
Specifies whether the unload status of system log files is to be checked. If unload status checking is not performed, system log information is not available for database recovery.	`pd_log_unload_check`
Specifies the size of the buffer to be used for system log input/output.	`pd_log_max_data_size`
Specifies the number of buffer sectors to be used for system log output.	`pd_log_write_buff_count`
Specifies the system log file record length.	`pd_log_rec_leng`
Specifies the number of buffer sectors to be used for rollback log input.	`pd_log_rollback_buff_count`
Specifies the number of records to be added to a system log file each time extension is triggered and an upper limit for extending the file size.	`pd_log_auto_expand_size`
Specifies the name of a file group for system log files.	`pdlogadfg -d sys`
Specifies the name of the system log file to be allocated to a file group.	`pdlogadpf -d sys`

(29) Operands related to synchronization point dump files

Operand explanation	Operand name
Specifies whether to use dual synchronization point dump files.	`pd_spd_dual`
Specifies whether a message is to be output when a synchronization point dump is completed.	`pd_spd_assurance_msg`
Specifies the number of guaranteed valid generations.	`pd_spd_assurance_count`
Specifies whether the reduced mode of operation is to be used.	`pd_spd_reduced_mode`
Specifies whether a reserved file is to be opened automatically.	`pd_spd_reserved_file_auto_open`
Specifies the size of the buffer to be used for synchronization point dump input/output.	`pd_spd_max_data_size`
Specifies the synchronization point dump collection interval.	`pd_log_sdinterval`
Specifies the name of a file group for a synchronization point dump file.	`pdlogadfg -d spd`
Specifies the name of a synchronization point dump file to be allocated to a file group.	`pdlogadpf -d spd`

(30) Operands related to status files

Operand explanation	Operand name
Specify the names of unit status files.	`pd_syssts_file_name_1-7`
Specifies the action to be taken when an error is detected in a unit status file during unit startup.	`pd_syssts_initial_error`
Specifies whether single operation of unit status files is to be used.	`pd_syssts_singleoperation`
Specifies the name of the unit status file to be used as the current file.	`pd_syssts_last_active_file`
During a restart, specifies the unit status file that was normal during the previous operation.	`pd_syssts_last_active_side`
Specify the names of server status files.	`pd_sts_file_name_1-7`
Specifies the action to be taken when an error is detected in a server status file during server startup.	`pd_sts_initial_error`
Specifies whether single operation of server status files is to be used.	`pd_sts_singleoperation`
Specifies the name of the server status file to be used as the current file.	`pd_sts_last_active_file`
During a restart, specifies the server status file that was normal during the previous operation.	`pd_sts_last_active_side`

(31) Operand related to message log file

Operand explanation	Operand name
Specifies a message log output destination unit.	`pd_mlg_msg_log_unit`
Specifies the maximum size of the message log file.	`pd_mlg_file_size`

(32) Operands related to statistical information

Operand explanation	Operand name
Specifies whether collection of statistical information is to begin at the time of HiRDB startup.	`pd_statistics`
Specifies the maximum size of the statistics log file.	`pd_stj_file_size`
Specifies the statistics log buffer size.	`pd_stj_buff_size`
Specifies the timing for collecting statistics on an SQL object buffer.	`pd_sqlobject_stat_timing`
Specifies the statistical information that is to be output, beginning at the time of HiRDB startup.	`pdstbegin`
Specifies that collection of statistical information related to CONNECT and DISCONNECT is to begin at the time of HiRDB startup.	`pdhibegin`

(33) Operands related to RPC trace information

Operand explanation	Operand name
Specifies whether RPC trace information is to be collected.	`pd_rpc_trace`
Specifies the name of the file in which RPC trace information is to be collected.	`pd_rpc_trace_name`
Specifies the size of the file in which RPC trace information is to be collected.	`pd_rpc_trace_size`

(34) Operands related to performance trace information

Operand explanation	Operand name
Specifies whether performance trace information is to be collected.	pd_prf_trace
Specifies the performance trace information collection level.	pd_prf_level
Specifies the number of performance trace information file generations.	pd_prf_file_count
Specifies the size of each performance trace information file.	pd_prf_file_size

(35) Operands related to troubleshooting information

Operand explanation	Operand name
Specifies whether to output troubleshooting information.	pd_cancel_dump
Specifies whether to collect the following types of troubleshooting information when the client maximum wait time (value of the PDCWAITTIME operand in the client environment definition) is exceeded during transaction execution.	pd_client_waittime_over_abort
Specifies whether to limit the units for which a shared memory dump can be output. This is the dump that is output during transaction execution when the client's maximum wait time (the value of the PDCWAITTIME operand in the client environment definition) is exceeded.	pd_clt_waittime_over_dump_level
Specifies the amount of time during which to suppress re-outputting the troubleshooting information.	pd_dump_suppress_watch_time
Specifies whether the network information portion of troubleshooting information is to be collected when a HiRDB process or HiRDB (unit) terminates abnormally.	pd_debug_info_netstat
Specifies the interval at which to delete the troubleshooting information files (files under %PDDIR%\spool) and the temporary work files (files under %PDDIR%\tmp) that are output by HiRDB.	pd_spool_cleanup_interval
Specifies the files to be deleted by the pd_spool_cleanup_interval operand.	pd_spool_cleanup_interval_level
Specifies whether to delete during the startup of HiRDB the troubleshooting information files (files under %PDDIR%\spool) that are output by HiRDB.	pd_spool_cleanup
Specifies the troubleshooting information files to be deleted by the pd_spool_cleanup operand.	pd_spool_cleanup_level
Specifies the maximum number of module trace records to be stored.	pd_module_trace_max
Specifies how to acquire the time to be output in module traces.	pd_module_trace_timer_level
Specifies the maximum number of communication traces to be stored.	pd_pth_trace_max

(36) Operands related to the BES connection holding facility

Operand explanation	Operand name
Specifies whether to use the BES connection holding facility.	pd_bes_connection_hold
Specifies the BES connection holding period.	pd_bes_conn_hold_trn_interval

(37) Operand related to the facility for predicting reorganization time

Operand explanation	Operand name
Specifies whether to use the facility for predicting reorganization time.	`pd_rorg_predict`

(38) Operands related to security

Operand explanation	Operand name
Specifies whether to begin collecting an audit trail when HiRDB (a unit for a HiRDB/Parallel Server) is started.	`pd_audit`
Specifies an absolute path name for the name of the HiRDB file system area for an audit trail file.	`pd_aud_file_name`
Specifies the maximum size (in megabytes) for audit trail files.	`pd_aud_max_generation_size`
Specifies the maximum number of (number of generations of) audit trail files to be created inside the HiRDB file system area for audit trail files.	`pd_aud_max_generation_num`
Specifies the process to be performed by HiRDB when no swappable audit trail file is available.	`pd_aud_no_standby_file_opr`
Specifies the size (in bytes) of the buffer to be used for asynchronously outputting an audit trail.	`pd_aud_async_buff_size`
Specifies the number of buffer sectors to be used for asynchronously outputting an audit trail.	`pd_aud_async_buff_count`
Specifies the retry interval for monitoring for a buffer to be used for asynchronous output of an audit trail in order to acquire an audit trail when all buffers are in use.	`pd_aud_async_buff_retry_intvl`
Specifies the size in bytes of SQL statement output to the audit trail when the security audit facility is being used.	`pd_aud_sql_source_size`
Specifies the size in bytes of SQL data output to the audit trail when the security audit facility is being used.	`pd_aud_sql_data_size`
When the number of unswappable audit trail files reaches or exceeds the warning value, a warning message is issued. For this operand, specify the warning value as a percentage of the maximum audit trail file count specified in the `pd_aud_max_generation_num` operand.	`pd_aud_file_wrn_pnt`
Specifies whether the facility for automatically loading audit trail table data is to be used.	`pd_aud_auto_loading`
Defines environment information for the database creation utility (`pdload`) activated by the facility for automatically loading audit trail table data.	`pdaudload`
Specifies the buffer size to be used to store information for the security audit facility.	`pd_audit_def_buffer_size`
Specifies all host names used on the network that constitutes the HiRDB server.	`pd_security_host_group`

(39) Operands related to system switchover facility

Operand explanation	Operand name
Specifies whether the system switchover facility is to be used.	`pd_ha`
Specifies whether IP addresses are to be inherited when the system switchover facility is used.	`pd_ha_ipaddr_inherit`
Specifies that the system switchover facility is not to be applied to the unit.	`pd_ha_unit`

Operand explanation	Operand name
Specifies whether to run the system switchover facility in the monitor mode or server mode.	`pd_ha_acttype`
Specifies whether to use user server hot standby.	`pd_ha_server_process_standby`
Specifies whether to use the rapid system switchover facility, standby-less system switchover (1:1) facility, or standby-less system switchover (effects distributed) facility.	`pd_ha_agent`
Specifies the maximum number of guest BESs that operate as running systems inside the unit when the standby-less system switchover (effects distributed) facility is used.	`pd_ha_max_act_guest_servers`
Specifies the maximum number of user server processes to be started inside a unit when the standby-less system switchover (effects distributed) facility is used.	`pd_ha_max_server_process`
Specifies the combined total of the number of host BESs inside the unit and the number of resident processes for the guest BESs after these guest BESs have been accepted when the standby-less system switchover (effects distributed) facility is used.	`pd_ha_process_count`
Specifies a maximum amount of time to wait until the running server's resources are activated at the time of unit startup when the standby-less system switchover (effects distributed) facility is used.	`pd_ha_resource_act_wait_time`
Specifies whether to use the transaction queuing facility. Also specifies the processing that takes place when the number of connections to the HiRDB server exceeds the maximum number of concurrent connections (value specified by the `pd_max_users` operand) during system switchover.	`pd_ha_transaction`
Specifies the transaction queuing wait time when the transaction queuing facility is used.	`pd_ha_trn_queuing_wait_time`
Specifies the upper limit for retrying transaction start requests after system switchover occurs and before the standby unit restarts.	`pd_ha_trn_restart_retry_time`
Specifies whether to switch the system without waiting for internal termination processing of HiRDB when internal termination processing of HiRDB (or a unit for a HiRDB/Parallel Server) during system switchover has exceeded the server failure monitoring time.	`pd_ha_switch_timeout`
Specifies whether to suppress switchover when a unit terminates abnormally during forced stop processing in the primary system when running switchovers in the server mode.	`pd_deter_restart_on_stop_fail`
Specifies whether to wait for the completion of start processing of other units before switching the system for the system manager unit.	`pd_ha_mgr_rerun`
Defines an HA group.	`pdhagroup`
Specifies that the system switchover facility is not to be applied to the unit.	`pdunit`
Specifies the server machine to be used for executing the servers that comprise a HiRDB system and how each server machine is to be used.	`pdstart`
Specifies the name of the standard host of the primary system.	`pd_hostname`

(40) Operands related to HiRDB Datareplicator

Operand explanation	Operand name
Specifies whether the HiRDB Datareplicator linkage facility is to used from the time of HiRDB startup.	`pd_rpl_init_start`
Specifies the unit for applying transactions when the HiRDB Datareplicator linkage facility is used.	`pd_rpl_reflect_mode`

Operand explanation	Operand name
Specifies the action to be taken when a swap request is received when not all of the system log files can be created at the swapping destination because system log extraction by HiRDB Datareplicator using the HiRDB Datareplicator linkage facility has not been completed.	`pd_log_rpl_no_standby_file_opr`
Specifies the extracted side HiRDB Datareplicator directory when the HiRDB Datareplicator linkage facility is used.	`pd_rpl_hdepath`
Specifies the facility to be run by the source HiRDB when the HiRDB Datareplicator linkage facility is used.	`pd_rpl_func_control`

(41) Operands related to linkage to JP1

Operand explanation	Operand name
Specifies whether HiRDB events are to be registered into JP1.	`pd_jp1_use`
Specifies the HiRDB event types that are to be registered into JP1.	`pd_jp1_event_level`
Specifies whether to register message-outputting events into JP1.	`pd_jp1_event_msg_out`

(42) Operand related to OLTP

Operand explanation	Operand name
Specifies whether to use the holdable cursor facility in a UAP under the OLTP environment.	`pd_oltp_holdcr`

(43) Operand related to date and time

Operand explanation	Operand name
Specifies whether leap seconds are to be used.	`pd_leap_second`
Specifies the time zone.	`TZ`

(44) Operand related to the message output suppression facility

Operand explanation	Operand name
Controls output or suppresses output of messages that HiRDB outputs to the event log.	`pdmlgput`

(45) Operand to client group

Operand explanation	Operand name
Specifies the number of users who are guaranteed connection when the connection frame guarantee facility for a client group is used.	`pdcltgrp`

(46) Operands related to plug-in

Operand explanation	Operand name
Specifies the name of a plug-in to be used.	`pdplugin`

Operand explanation	Operand name
Specifies the size of the shared memory to be used by a plug-in.	`pdplgprm`
Specifies the directory in which the index information file is to be created when delayed batch creation of plug-in indexes is to be performed.	`pd_plugin_ixmk_dir`

(47) Operands related to version upgrade

Operand explanation	Operand name
Specifies whether the `pdvrup` command for executing HiRDB version upgrading is to be started automatically.	`pd_auto_vrup`
Specifies whether to keep the default values of the system definition operands of HiRDB Version 6 or earlier.	`pd_sysdef_default_option`

(48) Operands related to communication processing

Operand explanation	Operand name
Specifies the port number for the scheduler process.	`pd_service_port`
Specifies how addresses are to be resolved in communication between units that is handled by the system server.	`pd_name_fixed_port_lookup`
Specifies the port number of the scheduler process when `Y` is specified in the `pd_name_fixed_port_lookup` operand.	`pd_scd_port`
Specifies the port number of the transaction server process when `Y` is specified in the `pd_name_fixed_port_lookup` operand.	`pd_trn_port`
Specifies the port number of the message log server process when `Y` is specified in the `pd_name_fixed_port_lookup` operand.	`pd_mlg_port`
Specifies the port number of the unit monitoring process when `Y` is specified in the `pd_name_fixed_port_lookup` operand.	`pd_alv_port`
Specifies the network to be used by the HiRDB server to communicate with HiRDB clients.	`pd_change_clt_ipaddr`
Specifies the range of port numbers to be used for communication by the HiRDB server.	`pd_registered_port`
Specifies whether checking is to be performed for overlapping port numbers in the ranges of port numbers specified in the `pd_registered_port` operand and in the port numbers registered in `%windir%\system32\drivers\etc\services`.	`pd_registered_port_check`
Specifies a target range for the HiRDB reserved port facility.	`pd_registered_port_level`
Specifies the number of times process-to-process communication can be attempted.	`pd_ipc_send_retrycount`
Specifies the sleep time between process-to-process send retries.	`pd_ipc_send_retrysleeptime`
Specifies the number of times server-to-server send can be performed before send is terminated.	`pd_ipc_send_count`
Specifies the number of times server-to-server receive can be performed before receive is terminated.	`pd_ipc_recv_count`
Specifies whether to establish connections in the non-block mode for HiRDB server-to-server communication.	`pd_ipc_conn_nblock`

Operand explanation	Operand name
Specifies the retry interval and monitoring time when establishing a connection in the non-block mode for HiRDB server-to-server communication.	`pd_ipc_conn_nblock_time`
Specifies the retry interval when establishing a connection for sending data between HiRDB servers.	`pd_ipc_conn_interval`
Specifies the number of retries when establishing a connection for sending data between HiRDB servers.	`pd_ipc_conn_count`
Specifies the size of the send/receive buffer to be used for communication with HiRDB.	`pd_inet_unix_bufmode`
Specifies the maximum size of the send/receive buffer to be used for server-to-server communication (TCP INET domain).	`pd_ipc_inet_bufsize`
Specifies whether to establish connections in the non-block mode for communication with HiRDB clients.	`pd_ipc_clt_conn_nblock`
Specifies the monitoring interval when establishing a connection with a HiRDB client in the non-block mode.	`pd_ipc_clt_conn_nblock_time`
Specifies the maximum size of the send/receive buffer to be used for communication (TCP INET domain) with HiRDB clients.	`pd_tcp_inet_bufsize`
Specifies whether the send/receive buffer is to be set in a `listen` socket when the `listen` socket for a TCP INET domain that is used for HiRDB communication processing is created.	`pd_listen_socket_bufset`
Specifies the size of the communication buffer used by utility processes.	`pd_utl_buff_size`
Specifies the size of the send/receive buffer to be used for communication that uses the `pd_utl_buff_size` operand.	`pd_inet_unix_utl_bufmode`
Specifies the size of the data per access when a utility accesses a file.	`pd_utl_file_buff_size`
Specifies the size of the communication buffer used for transferring retrieved data among HiRDB servers.	`pd_sql_send_buff_size`

(49) Operands related to Java

Operand explanation	Operand name
Specifies Java virtual option startup options.	`pd_java_option`
Specifies in bytes the stack size to be used by an external Java routine.	`pd_java_routine_stack_size`
Specifies the name of the directory for storing JAR files used by Java stored procedures or Java stored functions.	`pd_java_archive_directory`
Specifies the class path to be used by a Java virtual machine.	`pd_java_classpath`
Specifies the root directory of the Java Runtime Environment as an absolute path name.	`pd_java_runtimepath`
Specifies the directory that stores the library of the Java virtual machine as a relative path name to the Java Runtime Environment root directory.	`pd_java_libpath`
Specifies the file to which the standard output and standard error output are to be output in a Java virtual machine.	`pd_java_stdout_file`
Specifies whether to shut down the process that started the Java Virtual Machine at particular triggers.	`pd_java_castoff`

(50) Operands related to external C stored routines

Operand explanation	Operand name
Specifies the directory that stores the C library files used by external C stored routines.	`pd_c_library_directory`

(51) Operands related to SQL runtime warning output facility

Operand explanation	Operand name
Specifies the condition for outputting SQL runtime warning information.	`pd_cwaittime_wrn_pnt`
Specifies the output destination directory for the SQL runtime warning information file.	`pd_cwaittime_report_dir`
Specifies the maximum size for the SQL runtime warning information file.	`pd_cwaittime_report_size`

(52) Operands related to local buffers

Operand explanation	Operand name
Specifies the action to be taken by the UAP when the RDAREA or index to be accessed using a local buffer is being used by another user.	`pd_uap_wait`
Specifies the local buffer used by a UAP.	`pdlbuffer`

(53) Operands related to character encoding

Operand explanation	Operand name
Specifies the maximum number of bytes to be used to represent a single character.	`pd_substr_length`

(54) Operands related to modifying work file output directories

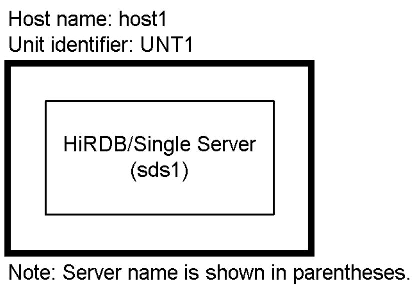
Operand explanation	Operand name
Specifies the target directory to which HiRDB is to output work files.	`pd_tmp_directory`

B. Examples of Definitions

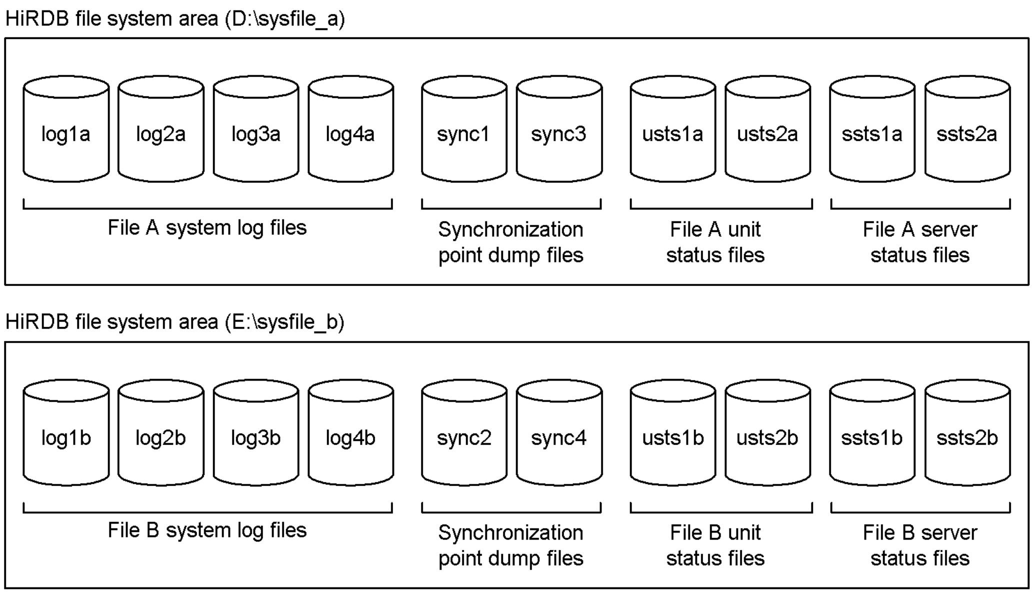
B.1 HiRDB/Single Server

This example creates system definitions for a HiRDB/Single Server. The system configuration is shown as follows:

Unit configuration



System files configuration



(1) System common definition

```
set pd_system_id = PDB1                                    1
set pd_name_port = 20001                                   2
set pd_master_file_name = "C:\rdarea\mast\mast01"          3
set pd_max_users = 100                                     4
set pd_max_server_process = 220                            5
set pd_max_access_tables = 50                              6
set pd_sql_object_cache_size = 3000                        7
set pd_max_rdarea_no = 200                                 8
set pd_max_file_no = 600                                   9
set pd_optimize_level = "PRIOR_NEST_JOIN", \               10
    "PRIOR_OR_INDEXES","DETER_AND_INDEXES", \
    "RAPID_GROUPING","DETER_WORK_TABLE_FOR_UPDATE", \
    "APPLY_ENHANCED_KEY_COND"
set pd_additional_optimize_level = "COST_BASE_2"           11
pdunit -x host1 -u UNT1 -d "C:\HiRDB"                      12
pdstart -t SDS -s sds1 -u UNT1                             13
pdbuffer -a gbuf01 -r RDMAST,RDDIC,RDDIR -n 1000           14
pdbuffer -a gbuf02 -r RDAREA1,RDAREA2,RDAREA3 -n 1000      15
pdbuffer -a gbuf03 -r RDAREA4,RDAREA5,RDAREA6 -n 1000         16
pdbuffer -a gbuf04 -o -n 1000                                 17
putenv SHMMAX 16                                              18
```

Explanation

1. Specifies the HiRDB identifier.
2. Specifies the HiRDB port number.
3. Specifies the first HiRDB file in the master directory RDAREA.
4. Specifies the maximum number of concurrent connections.
5. Specifies the maximum number of server processes that can be activated concurrently.
6. Specifies the maximum number of base tables that can be accessed concurrently.
7. Specifies the SQL object buffer size.
8. Specifies the maximum number of RDAREAs allowed.
9. Specifies the maximum number of HiRDB files that comprise an RDAREA.
10. Specifies SQL optimization options.
11. Specifies an SQL extension optimizing option.
12. Specifies the unit configuration:
    `-x`: Host name
    `-u`: Unit identifier
    `-d`: HiRDB directory name
13. Specifies the server configuration:
    `-t`: Server type (SDS: single server)
    `-s`: Server name
    `-u`: Unit identifier
14. Allocates the master directory RDAREA, data dictionary RDAREA, and data directory RDAREA to a global buffer.
15. Allocates user RDAREAs (RDAREA1-RDAREA3) to a global buffer.
16. Allocates user RDAREAs (RDAREA4-RDAREA6) to a global buffer.
17. Allocates other RDAREAs to a global buffer.
18. Specifies the maximum number of shared memory segments.

(2) Unit control information definition

```
set pd_unit_id = UNT1                                                1
set pd_syssts_file_name_1 = "untsts1","D:\sysfile_a\usts1a",\        2
                                      "E:\sysfile_b\usts1b"
set pd_syssts_file_name_2 = "untsts2","D:\sysfile_a\usts2a",\
                                      "E:\sysfile_b\usts2b"
```

Explanation

1. Specifies the unit identifier.
2. Specifies the configuration of the unit status files.

(3) Single server definition

```
set pd_sds_shmpool_size = 10000                                       1
set pd_log_dual = Y                                                   2
set pd_sts_file_name_1 = "svrsts1","D:\sysfile_a\ssts1a",\            3
                                   "E:\sysfile_b\ssts1b"
set pd_sts_file_name_2 = "svrsts2","D:\sysfile_a\ssts2a",\
                                   "E:\sysfile_b\ssts2b"
pdwork -v "C:\work01","C:\work02"                                     4
pdlogadfg -d sys -g log1 ONL                                          5
pdlogadfg -d sys -g log2 ONL
pdlogadfg -d sys -g log3 ONL
pdlogadfg -d sys -g log4 ONL
pdlogadpf -d sys -g log1 -a "D:\sysfile_a\log1a" -b "E:\sysfile_b\log1b"
pdlogadpf -d sys -g log2 -a "D:\sysfile_a\log2a" -b "E:\sysfile_b\log2b"
pdlogadpf -d sys -g log3 -a "D:\sysfile_a\log3a" -b "E:\sysfile_b\log3b"
pdlogadpf -d sys -g log4 -a "D:\sysfile_a\log4a" -b "E:\sysfile_b\log4b"
pdlogadfg -d spd -g sync1 ONL                                             6
pdlogadfg -d spd -g sync2 ONL
pdlogadfg -d spd -g sync3 ONL
pdlogadfg -d spd -g sync4 ONL
pdlogadpf -d spd -g sync1 -a "D:\sysfile_a\sync1"
pdlogadpf -d spd -g sync2 -a "E:\sysfile_b\sync2"
pdlogadpf -d spd -g sync3 -a "D:\sysfile_a\sync3"
pdlogadpf -d spd -g sync4 -a "E:\sysfile_b\sync4"
```

Explanation

1. Specifies the size of the shared memory to be used by the single server.
2. Specifies use of dual system log files.
3. Specifies the configuration of the server status files.
4. Specifies HiRDB file system areas for work table files.
5. Specifies the system log files configuration.
6. Specifies the synchronization point dump files configuration.

(4) UAP environment definition

```
set pd_uap_wait = Y                                        1
pdlbuffer -a localbuf1 -r RDAREA10 -n 1000 -p 16           2
pdlbuffer -a localbuf2 -r RDAREA11,RDAREA12 -n 1000        3
pdlbuffer -a localbuf3 -i USER01.INDX01 -n 1000            4
```

Explanation

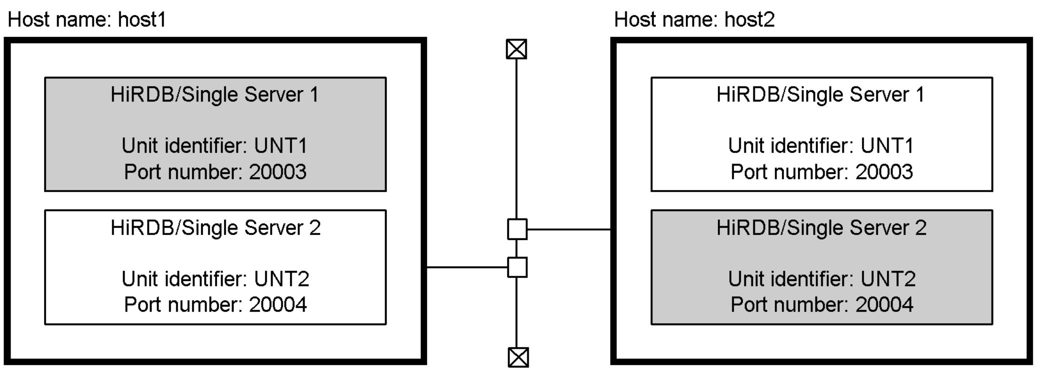
1. Specifies the action to be taken by the UAP when the RDAREA or index to be accessed using a local buffer is being used by another user.
2. Allocates a local buffer to a user RDAREA (`RDAREA10`).
3. Allocates a local buffer to user RDAREAs (`RDAREA11` and `RDAREA12`).
4. Allocates a local buffer to an index (`INDX01`).

B.2 HiRDB/Single Server: system switchover facility being used

This example creates system definitions for a HiRDB/Single Server. The system configuration is shown as follows:

- This configuration is based on mutual system switchover between HiRDB/Single Server 1 and HiRDB/Single Server 2.
- The system switchover facility does not inherit IP addresses.

This example explains only the system common definition, unit control definition, and their related operands.



Note: Shading indicates the primary HiRDB systems.

(1) System common definition of HiRDB/Single Server 1

```
set pd_system_id = SDS1                                     1
set pd_name_port = 20003                                    2
         :
         :
set pd_ha = use                                             3
set pd_ha_ipaddr_inherit = N                                4
         :
         :
pdunit -x host1 -u UNT1 -d "C:\HiRDB1" -c host2             5
pdstart -t SDS -s sds01 -u UNT1                             6
```

Explanation

1. Specifies the HiRDB identifier.
2. Specifies the HiRDB port number.
3. Specifies use of the system switchover facility.
4. Specifies that IP addresses are not inherited.
5. Specifies the unit configuration of the HiRDB/Single Server 1:
   `-x`: Host name.
   `-u`: Unit identifier
   `-d`: HiRDB directory name
   `-c`: Host name of the secondary system
6. Specifies the server configuration of the HiRDB/Single Server 1:
   `-t`: Server type
   `-s`: Server name
   `-u`: Unit identifier

(2) Unit control information definition of HiRDB/Single Server 1

```
set pd_unit_id = UNT1                                        1
         :
         :
set pd_hostname = host1                                      2
```

Explanation

1. Specifies the unit identifier.
2. Specifies the host name of the primary system.

(3) System common definition of HiRDB/Single Server 2

```
set pd_system_id = SDS2                                   1
set pd_name_port = 20004                                  2
         :
         :
set pd_ha = use                                           3
set pd_ha_ipaddr_inherit = N                              4
         :
         :
pdunit -x host2 -u UNT2 -d "C:\HiRDB2" -c host1           5
pdstart -t SDS -s sds02 -u UNT2                           6
```

Explanation

1. Specifies the HiRDB identifier.
2. Specifies the HiRDB port number.
3. Specifies use of the system switchover facility.
4. Specifies that IP addresses are not inherited.
5. Specifies the unit configuration of the HiRDB/Single Server 2:
   `-x`: Host name.
   `-u`: Unit identifier
   `-d`: HiRDB directory name
   `-c`: Host name of the secondary system
6. Specifies the server configuration of the HiRDB/Single Server 2:
   `-t`: Server type
   `-s`: Server name
   `-u`: Unit identifier

(4) Unit control information definition of HiRDB/Single Server 2

```
set pd_unit_id = UNT2                                     1
         :
         :
set pd_hostname = host2                                   2
```

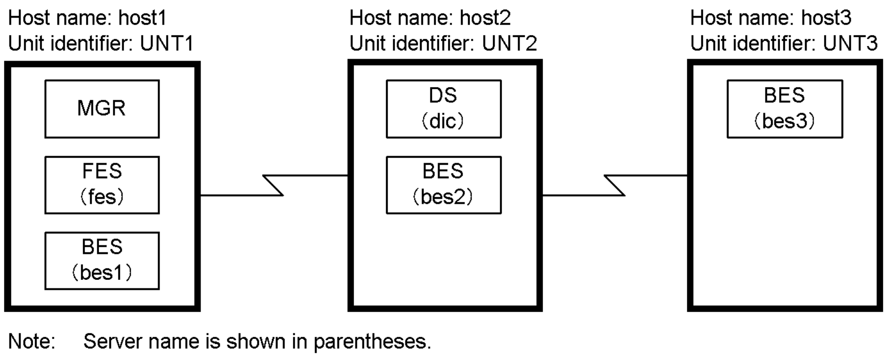
Explanation

1. Specifies the unit identifier.
2. Specifies the host name of the primary system.

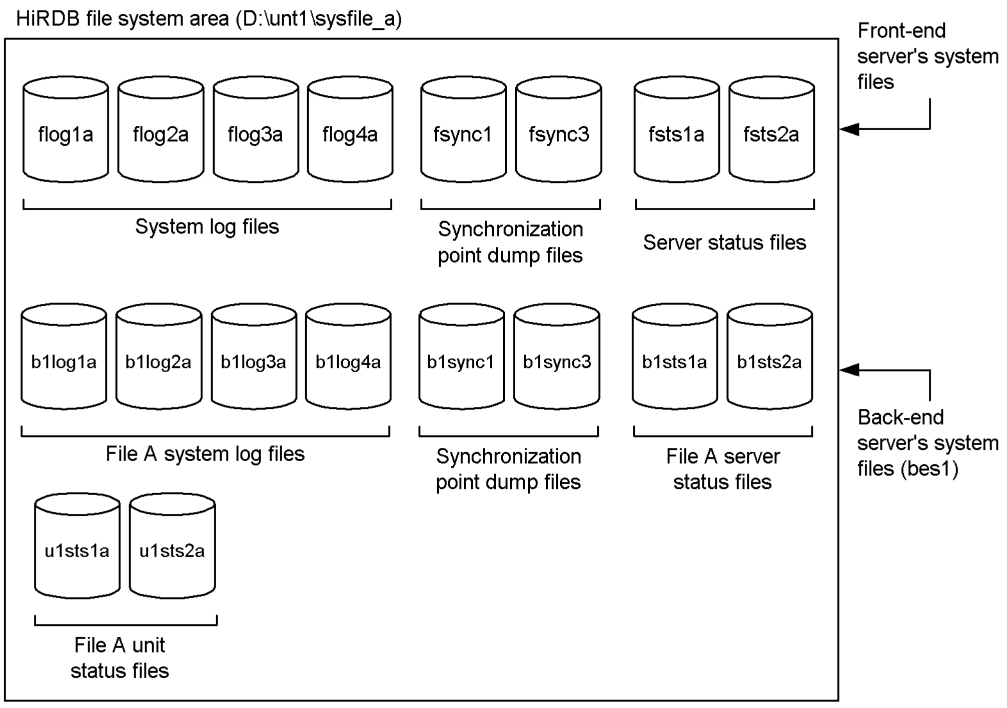
B.3 HiRDB/Parallel Server

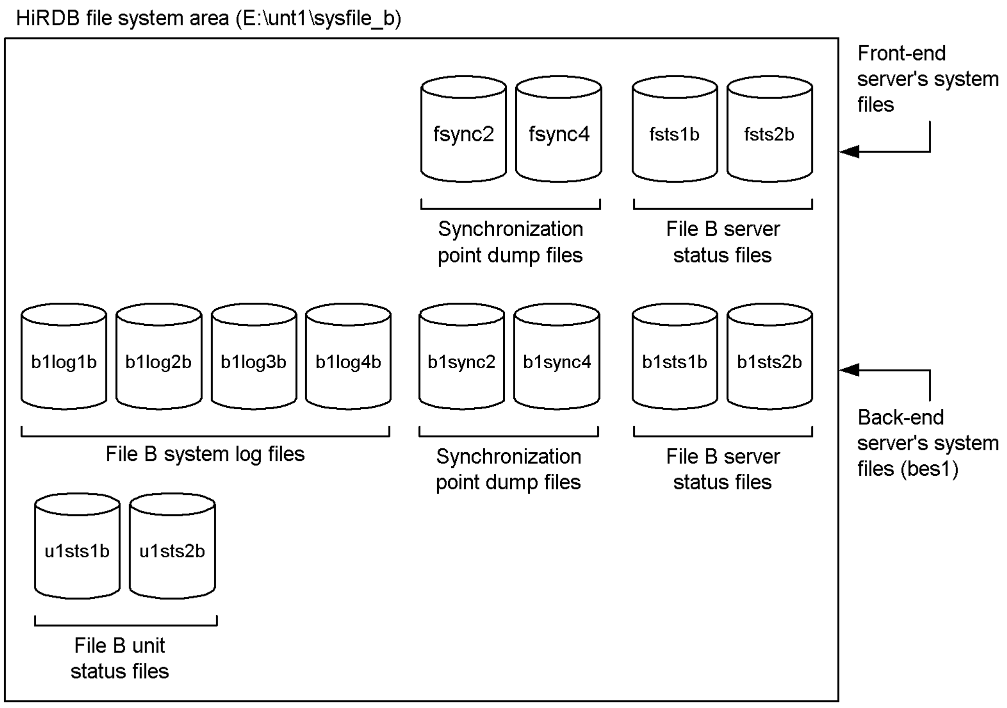
This example creates system definitions for HiRDB/Parallel Server. The system configuration is shown as follows:

Unit configuration

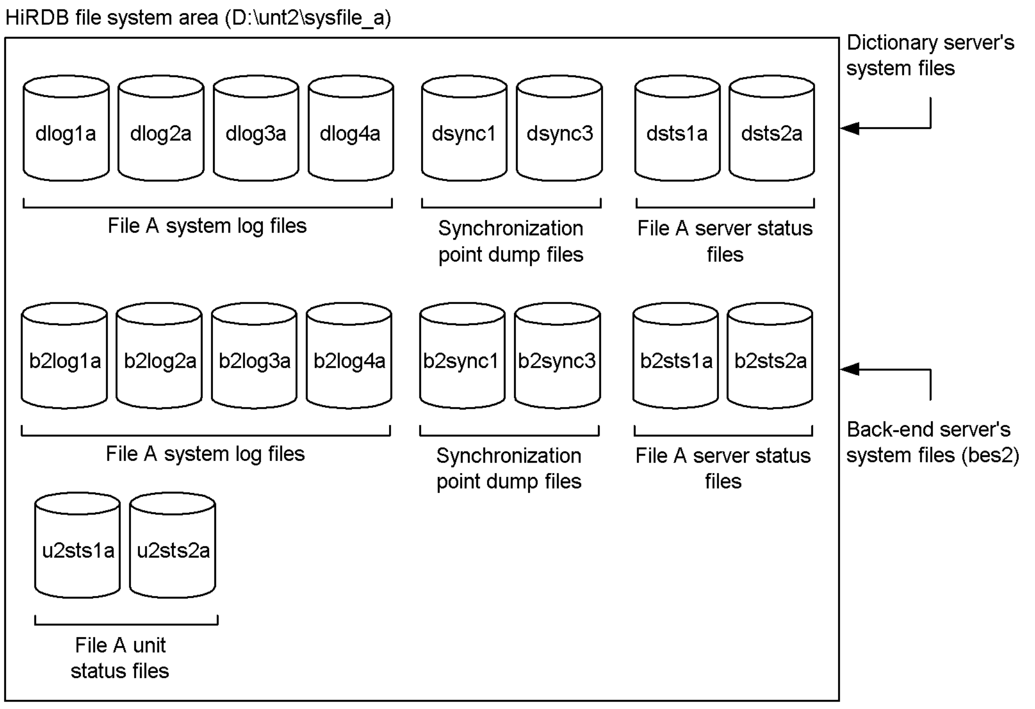


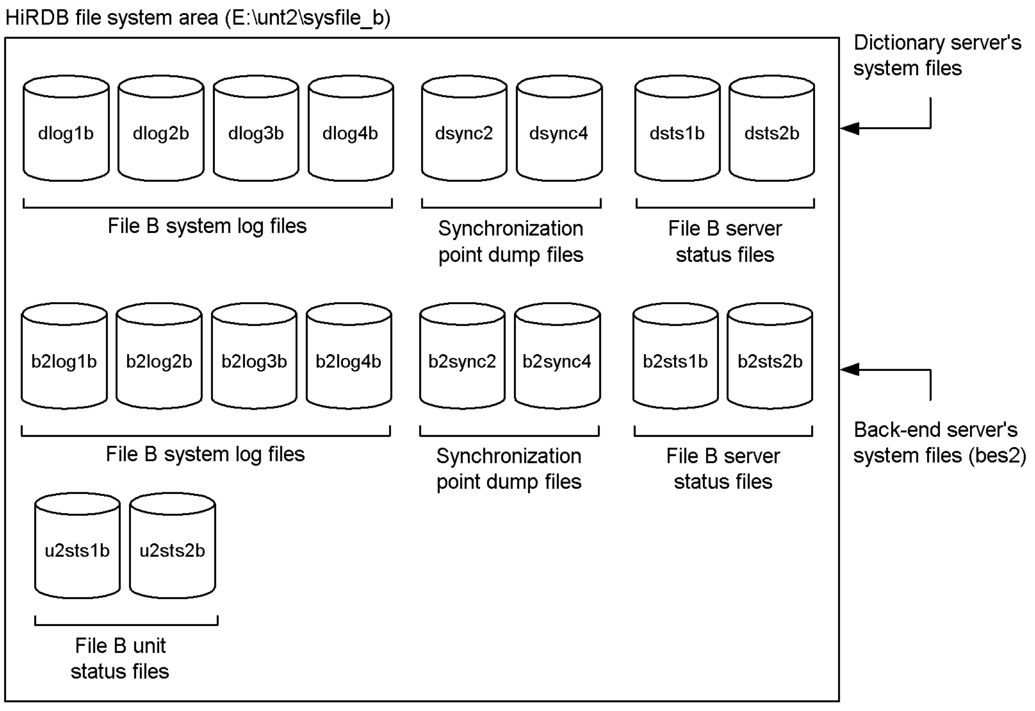
System files configuration for UNT1


HiRDB file system area (D:\unt1\sysfile_a)
flog1a
flog2a
flog3a
flog4a
fsync1
fsync3
fsts1a
fsts2a
System log files
Synchronization point dump files
Server status files
Front-end server's system files
b1log1a
b1log2a
b1log3a
b1log4a
b1sync1
b1sync3
b1sts1a
b1sts2a
File A system log files
Synchronization point dump files
File A server status files
Back-end server's system files (bes1)
u1sts1a
u1sts2a
File A unit status files

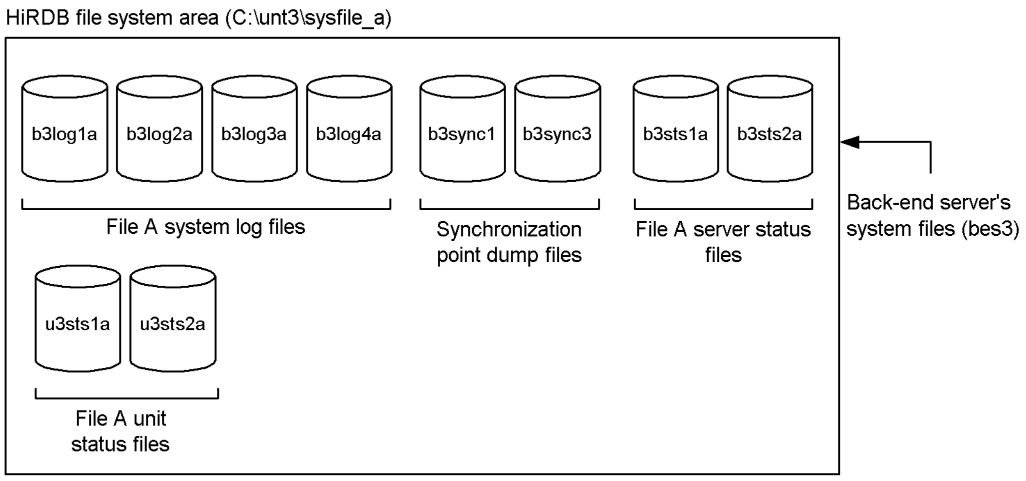

HiRDB file system area (E:\unt1\sysfile_b)
fsync2
fsync4
fsts1b
fsts2b
Synchronization point dump files
File B server status files
Front-end server's system files
b1log1b
b1log2b
b1log3b
b1log4b
b1sync2
b1sync4
b1sts1b
b1sts2b
File B system log files
Synchronization point dump files
File B server status files
Back-end server's system files (bes1)
u1sts1b
u1sts2b
File B unit status files

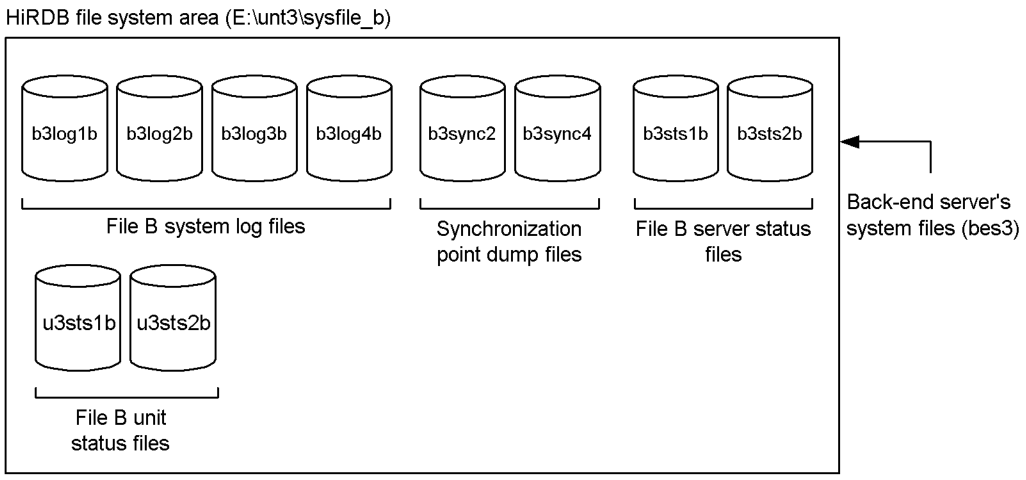
System files configuration for UNT2


HiRDB file system area (D:\unt2\sysfile_a)
Dictionary server's system files
dlog1a
dlog2a
dlog3a
dlog4a
dsync1
dsync3
dsts1a
dsts2a
File A system log files
Synchronization point dump files
File A server status files
b2log1a
b2log2a
b2log3a
b2log4a
b2sync1
b2sync3
b2sts1a
b2sts2a
File A system log files
Synchronization point dump files
File A server status files
Back-end server's system files (bes2)
u2sts1a
u2sts2a
File A unit status files


HiRDB file system area (E:\unt2\sysfile_b)
Dictionary server's system files
dlog1b
dlog2b
dlog3b
dlog4b
dsync2
dsync4
dsts1b
dsts2b
File B system log files
Synchronization point dump files
File B server status files
b2log1b
b2log2b
b2log3b
b2log4b
b2sync2
b2sync4
b2sts1b
b2sts2b
File B system log files
Synchronization point dump files
File B server status files
Back-end server's system files (bes2)
u2sts1b
u2sts2b
File B unit status files

System files configuration for UNT3


HiRDB file system area (C:\unt3\sysfile_a)
b3log1a
b3log2a
b3log3a
b3log4a
File A system log files
b3sync1
b3sync3
Synchronization point dump files
b3sts1a
b3sts2a
File A server status files
Back-end server's system files (bes3)
u3sts1a
u3sts2a
File A unit status files


HiRDB file system area (E:\unt3\sysfile_b)
b3log1b
b3log2b
b3log3b
b3log4b
File B system log files
b3sync2
b3sync4
Synchronization point dump files
b3sts1b
b3sts2b
File B server status files
Back-end server's system files (bes3)
u3sts1b
u3sts2b
File B unit status files

(1) System common definition

```
set pd_system_id = PDB1                                             1
set pd_name_port = 20001                                            2
set pd_master_file_name = "C:\rdarea\mast\mast01"                   3
set pd_max_users = 100                                              4
set pd_max_server_process = 520                                     5
set pd_max_access_tables = 50                                       6
set pd_sql_object_cache_size = 3000                                 7
set pd_max_rdarea_no = 200                                          8
set pd_max_file_no = 600                                            9
set pd_optimize_level = "PRIOR_NEST_JOIN","PRIOR_OR_INDEXES", \     10
    "SORT_DATA_BES","DETER_AND_INDEXES","RAPID_GROUPING", \
    "DETER_WORK_TABLE_FOR_UPDATE", \
    "APPLY_ENHANCED_KEY_COND"
set pd_additional_optimize_level = "COST_BASE_2"                    11
pdunit -x host1 -u UNT1 -d "C:\HiRDB"                               12
pdunit -x host2 -u UNT2 -d "C:\HiRDB"
pdunit -x host3 -u UNT3 -d "C:\HiRDB"
pdstart -t MGR -u UNT1                                              13
pdstart -t FES -s fes -u UNT1
pdstart -t DIC -s dic -u UNT2
pdstart -t BES -s bes1 -u UNT1
pdstart -t BES -s bes2 -u UNT2
pdstart -t BES -s bes3 -u UNT3
pdbuffer -a gbuf01 -r RDMAST,RDDIC,RDDIR -n 1000                    14
pdbuffer -a gbuf02 -r RDAREA1,RDAREA2,RDAREA3 -n 1000               15
pdbuffer -a gbuf03 -r RDAREA4,RDAREA5,RDAREA6 -n 1000               16
pdbuffer -a gbuf04 -o -n 1000                                       17
putenv SHMMAX 16                                                    18
```

Explanation

1. Specifies the HiRDB identifier.
2. Specifies the HiRDB port number.
3. Specifies the first HiRDB file in the master directory RDAREA.
4. Specifies the maximum number of concurrent connections.
5. Specifies the maximum number of server processes that can be activated concurrently.
6. Specifies the maximum number of base tables that can be accessed concurrently.
7. Specifies the SQL object buffer length.
8. Specifies the maximum number of RDAREAs allowed.
9. Specifies the maximum number of HiRDB files that comprise an RDAREA.
10. Specifies SQL optimization options.
11. Specifies an SQL extension optimizing option.
12. Specifies the configurations of the units in the HiRDB/Parallel Server:
    `-x`: Host name
    `-u`: Unit identifier
    `-d`: HiRDB directory name
13. Specifies the configurations of the servers in the HiRDB/Parallel Server:
    `-t`: Server type
    `-s`: Server name
    `-u`: Unit identifier
14. Allocates the master directory RDAREA, data dictionary RDAREA, and data directory RDAREA to a global buffer.
15. Allocates user RDAREAs (RDAREA1-RDAREA3) to a global buffer.
16. Allocates user RDAREAs (RDAREA4-RDAREA6) to a global buffer.
17. Allocates other RDAREAs to a global buffer.
18. Specifies the maximum number of shared memory segments.

(2) Unit control information definition of UNT1

```
set pd_unit_id = UNT1                                                    1
set pd_syssts_file_name_1 = "u1sts1","D:\unt1\sysfile_a\u1sts1a",\       2
                                     "E:\unt1\sysfile_b\u1sts1b"
set pd_syssts_file_name_2 = "u1sts2","D:\unt1\sysfile_a\u1sts2a",\
                                     "E:\unt1\sysfile_b\u1sts2b"
```

Explanation

1. Specifies the unit identifier.
2. Specifies the configuration of the unit status files.

(3) Unit control information definition of UNT2

```
set pd_unit_id = UNT2                                                     1
set pd_syssts_file_name_1 = "u2sts1","D:\unt2\sysfile_a\u2sts1a",\        2
                                     "E:\unt2\sysfile_b\u2sts1b"
set pd_syssts_file_name_2 = "u2sts2","D:\unt2\sysfile_a\u2sts2a",\
                                     "E:\unt2\sysfile_b\u2sts2b"
```

Explanation

1. Specifies the unit identifier.
2. Specifies the configuration of the unit status files.

(4) Unit control information definition of UNT3

```
set pd_unit_id = UNT3                                                  1
set pd_syssts_file_name_1 = "u3sts1","D:\unt3\sysfile_a\u3sts1a",\     2
                                     "E:\unt3\sysfile_b\u3sts1b"
set pd_syssts_file_name_2 = "u3sts2","D:\unt3\sysfile_a\u3sts2a",\
                                     "E:\unt3\sysfile_b\u3sts2b"
```

Explanation

1. Specifies the unit identifier.
2. Specifies the configuration of the unit status files.

(5) Front-end server definition

```
set pd_log_dual = N                                                   1
set pd_sts_file_name_1 = "fsts1","D:\unt1\sysfile_a\fsts1a",\         2
                                 "E:\unt1\sysfile_b\fsts1b"
set pd_sts_file_name_2 = "fsts2","D:\unt1\sysfile_a\fsts2a",\
                                 "E:\unt1\sysfile_b\fsts2b"
pdlogadfg -d sys -g flog1 ONL                                         3
pdlogadfg -d sys -g flog2 ONL
pdlogadfg -d sys -g flog3 ONL
pdlogadfg -d sys -g flog4 ONL
pdlogadpf -d sys -g flog1 -a "D:\unt1\sysfile_a\flog1a"
pdlogadpf -d sys -g flog2 -a "D:\unt1\sysfile_a\flog2a"
pdlogadpf -d sys -g flog3 -a "D:\unt1\sysfile_a\flog3a"
pdlogadpf -d sys -g flog4 -a "D:\unt1\sysfile_a\flog4a"
pdlogadfg -d spd -g fsync1 ONL                                        4
pdlogadfg -d spd -g fsync2 ONL
pdlogadfg -d spd -g fsync3 ONL
pdlogadfg -d spd -g fsync4 ONL
pdlogadpf -d spd -g fsync1 -a "D:\unt1\sysfile_a\fsync1"
pdlogadpf -d spd -g fsync2 -a "E:\unt1\sysfile_b\fsync2"
pdlogadpf -d spd -g fsync3 -a "D:\unt1\sysfile_a\fsync3"
pdlogadpf -d spd -g fsync4 -a "E:\unt1\sysfile_b\fsync4"
```

Explanation

1. Specifies that dual system log files will not be used.
2. Specifies the configuration of the server status files.
3. Specifies the configuration of the system log files.
4. Specifies the configuration of the synchronization point dump files.

(6) Dictionary server definition

```
set pd_dic_shmpool_size = 5000                                        1
set pd_log_dual = Y                                                   2
set pd_sts_file_name_1 = "dsts1","D:\unt2\sysfile_a\dsts1a",\         3
                                 "E:\unt2\sysfile_b\dsts1b"
set pd_sts_file_name_2 = "dsts2","D:\unt2\sysfile_a\dsts2a",\
                                 "E:\unt2\sysfile_b\dsts2b"
pdwork -v "C:\unt2\work01","C:\unt2\work02"                           4
pdlogadfg -d sys -g dlog1 ONL                                         5
pdlogadfg -d sys -g dlog2 ONL
pdlogadfg -d sys -g dlog3 ONL
pdlogadfg -d sys -g dlog4 ONL
pdlogadpf -d sys -g dlog1 -a "D:\unt2\sysfile_a\dlog1a"\
                          -b "E:\unt2\sysfile_b\dlog1b"
pdlogadpf -d sys -g dlog2 -a "D:\unt2\sysfile_a\dlog2a"\
                          -b "E:\unt2\sysfile_b\dlog2b"
pdlogadpf -d sys -g dlog3 -a "D:\unt2\sysfile_a\dlog3a"\
                          -b "E:\unt2\sysfile_b\dlog3b"
pdlogadpf -d sys -g dlog4 -a "D:\unt2\sysfile_a\dlog4a"\
                          -b "E:\unt2\sysfile_b\dlog4b"
pdlogadfg -d spd -g dsync1 ONL                                        6
pdlogadfg -d spd -g dsync2 ONL
pdlogadfg -d spd -g dsync3 ONL
pdlogadfg -d spd -g dsync4 ONL
pdlogadpf -d spd -g dsync1 -a "D:\unt2\sysfile_a\dsync1"
pdlogadpf -d spd -g dsync2 -a "E:\unt2\sysfile_b\dsync2"
pdlogadpf -d spd -g dsync3 -a "D:\unt2\sysfile_a\dsync3"
pdlogadpf -d spd -g dsync4 -a "E:\unt2\sysfile_b\dsync4"
```

Explanation

1. Specifies the size of the shared memory to be used by the dictionary server.
2. Specifies that dual system log files will be used.
3. Specifies the configuration of the server status files.
4. Specifies HiRDB file system areas for work table files.
5. Specifies the configuration of the system log files.
6. Specifies the configuration of the synchronization point dump files.

(7) Back-end server definition of bes1

```
set pd_bes_shmpool_size = 5000                                          1
set pd_log_dual = Y                                                     2
set pd_sts_file_name_1 = "b1sts1","D:\unt1\sysfile_a\b1sts1a",\         3
                                  "E:\unt1\sysfile_b\b1sts1b"
set pd_sts_file_name_2 = "b1sts2","D:\unt1\sysfile_a\b1sts2a",\
                                  "E:\unt1\sysfile_b\b1sts2b"
pdwork -v "C:\unt1\work01","C:\unt1\work02"                             4
pdlogadfg -d sys -g b1log1 ONL                                          5
pdlogadfg -d sys -g b1log2 ONL
pdlogadfg -d sys -g b1log3 ONL
pdlogadfg -d sys -g b1log4 ONL
pdlogadpf -d sys -g b1log1 -a "D:\unt1\sysfile_a\b1log1a"\
                           -b "E:\unt1\sysfile_b\b1log1b"
pdlogadpf -d sys -g b1log2 -a "D:\unt1\sysfile_a\b1log2a"\
                           -b "E:\unt1\sysfile_b\b1log2b"
pdlogadpf -d sys -g b1log3 -a "D:\unt1\sysfile_a\b1log3a"\
                           -b "E:\unt1\sysfile_b\b1log3b"
pdlogadpf -d sys -g b1log4 -a "D:\unt1\sysfile_a\b1log4a"\
                           -b "E:\unt1\sysfile_b\b1log4b"
pdlogadfg -d spd -g b1sync1 ONL                                         6
pdlogadfg -d spd -g b1sync2 ONL
pdlogadfg -d spd -g b1sync3 ONL
pdlogadfg -d spd -g b1sync4 ONL
pdlogadpf -d spd -g b1sync1 -a "D:\unt1\sysfile_a\b1sync1"
pdlogadpf -d spd -g b1sync2 -a "E:\unt1\sysfile_b\b1sync2"
pdlogadpf -d spd -g b1sync3 -a "D:\unt1\sysfile_a\b1sync3"
pdlogadpf -d spd -g b1sync4 -a "E:\unt1\sysfile_b\b1sync4"
```

Explanation

1. Specifies the size of the shared memory to be used by the back-end server (`bes1`).
2. Specifies that dual system log files will be used.
3. Specifies the configuration of the server status files.
4. Specifies HiRDB file system areas for work table files.
5. Specifies the configuration of the system log files.
6. Specifies the configuration of the synchronization point dump files.

(8) Back-end server definition of bes2

```
set pd_bes_shmpool_size = 5000                                           1
set pd_log_dual = Y                                                      2
set pd_sts_file_name_1 = "b2sts1","D:\unt2\sysfile_a\b2sts1a",\          3
                                  "E:\unt2\sysfile_b\b2sts1b"
set pd_sts_file_name_2 = "b2sts2","D:\unt2\sysfile_a\b2sts2a",\
                                  "E:\unt2\sysfile_b\b2sts2b"
pdwork -v "C:\unt2\work03","C:\unt2\work04"                              4
pdlogadfg -d sys -g b2log1 ONL                                           5
pdlogadfg -d sys -g b2log2 ONL
pdlogadfg -d sys -g b2log3 ONL
pdlogadfg -d sys -g b2log4 ONL
pdlogadpf -d sys -g b2log1 -a "D:\unt2\sysfile_a\b2log1a"\
                           -b "E:\unt2\sysfile_b\b2log1b"
pdlogadpf -d sys -g b2log2 -a "D:\unt2\sysfile_a\b2log2a"\
                           -b "E:\unt2\sysfile_b\b2log2b"
pdlogadpf -d sys -g b2log3 -a "D:\unt2\sysfile_a\b2log3a"\
                           -b "E:\unt2\sysfile_b\b2log3b"
pdlogadpf -d sys -g b2log4 -a "D:\unt2\sysfile_a\b2log4a"\
                           -b "E:\unt2\sysfile_b\b2log4b"
pdlogadfg -d spd -g b2sync1 ONL                                             6
pdlogadfg -d spd -g b2sync2 ONL
pdlogadfg -d spd -g b2sync3 ONL
pdlogadfg -d spd -g b2sync4 ONL
pdlogadpf -d spd -g b2sync1 -a "D:\unt2\sysfile_a\b2sync1"
pdlogadpf -d spd -g b2sync2 -a "E:\unt2\sysfile_b\b2sync2"
pdlogadpf -d spd -g b2sync3 -a "D:\unt2\sysfile_a\b2sync3"
pdlogadpf -d spd -g b2sync4 -a "E:\unt2\sysfile_b\b2sync4"
```

Explanation

1. Specifies the size of the shared memory to be used by the back-end server (`bes2`).
2. Specifies that dual system log files will be used.
3. Specifies the configuration of the server status files
4. Specifies HiRDB file system areas for work table files.
5. Specifies the configuration of the system log files.
6. Specifies the configuration of the synchronization point dump files.

(9) Back-end server definition of bes3

```
set pd_bes_shmpool_size = 5000                                             1
set pd_log_dual = Y                                                        2
set pd_sts_file_name_1 = "b3sts1","D:\unt3\sysfile_a\b3sts1a",\            3
                                  "E:\unt3\sysfile_b\b3sts1b"
set pd_sts_file_name_2 = "b3sts2","D:\unt3\sysfile_a\b3sts2a",\
                                  "E:\unt3\sysfile_b\b3sts2b"
pdwork -v "C:\unt3\work01","C:\unt3\work02"                                4
pdlogadfg -d sys -g b3log1 ONL                                             5
pdlogadfg -d sys -g b3log2 ONL
pdlogadfg -d sys -g b3log3 ONL
pdlogadfg -d sys -g b3log4 ONL
pdlogadpf -d sys -g b3log1 -a "D:\unt3\sysfile_a\b3log1a"\
                           -b "E:\unt3\sysfile_b\b3log1b"
pdlogadpf -d sys -g b3log2 -a "D:\unt3\sysfile_a\b3log2a"\
                           -b "E:\unt3\sysfile_b\b3log2b"
pdlogadpf -d sys -g b3log3 -a "D:\unt3\sysfile_a\b3log3a"\
                           -b "E:\unt3\sysfile_b\b3log3b"
pdlogadpf -d sys -g b3log4 -a "D:\unt3\sysfile_a\b3log4a"\
                           -b "E:\unt3\sysfile_b\b3log4b"
pdlogadfg -d spd -g b3sync1 ONL                                            6
pdlogadfg -d spd -g b3sync2 ONL
pdlogadfg -d spd -g b3sync3 ONL
pdlogadfg -d spd -g b3sync4 ONL
pdlogadpf -d spd -g b3sync1 -a "D:\unt3\sysfile_a\b3sync1"
pdlogadpf -d spd -g b3sync2 -a "E:\unt3\sysfile_b\b3sync2"
pdlogadpf -d spd -g b3sync3 -a "D:\unt3\sysfile_a\b3sync3"
pdlogadpf -d spd -g b3sync4 -a "E:\unt3\sysfile_b\b3sync4"
```

Explanation

1. Specifies the size of the shared memory to be used by the back-end server (`bes3`).
2. Specifies that dual system log files will be used.
3. Specifies the configuration of the server status files.
4. Specifies HiRDB file system areas for work table files.
5. Specifies the configuration of the system log files.
6. Specifies the configuration of the synchronization point dump files.

(10) UAP environment definition

```
set pd_uap_wait = Y                                    1
pdlbuffer -a localbuf1 -r RDAREA10 -n 1000 -p 16       2
pdlbuffer -a localbuf2 -r RDAREA11,RDAREA12 -n 1000        3
pdlbuffer -a localbuf3 -i USER01.INDX01 -n 1000            4
```

Explanation

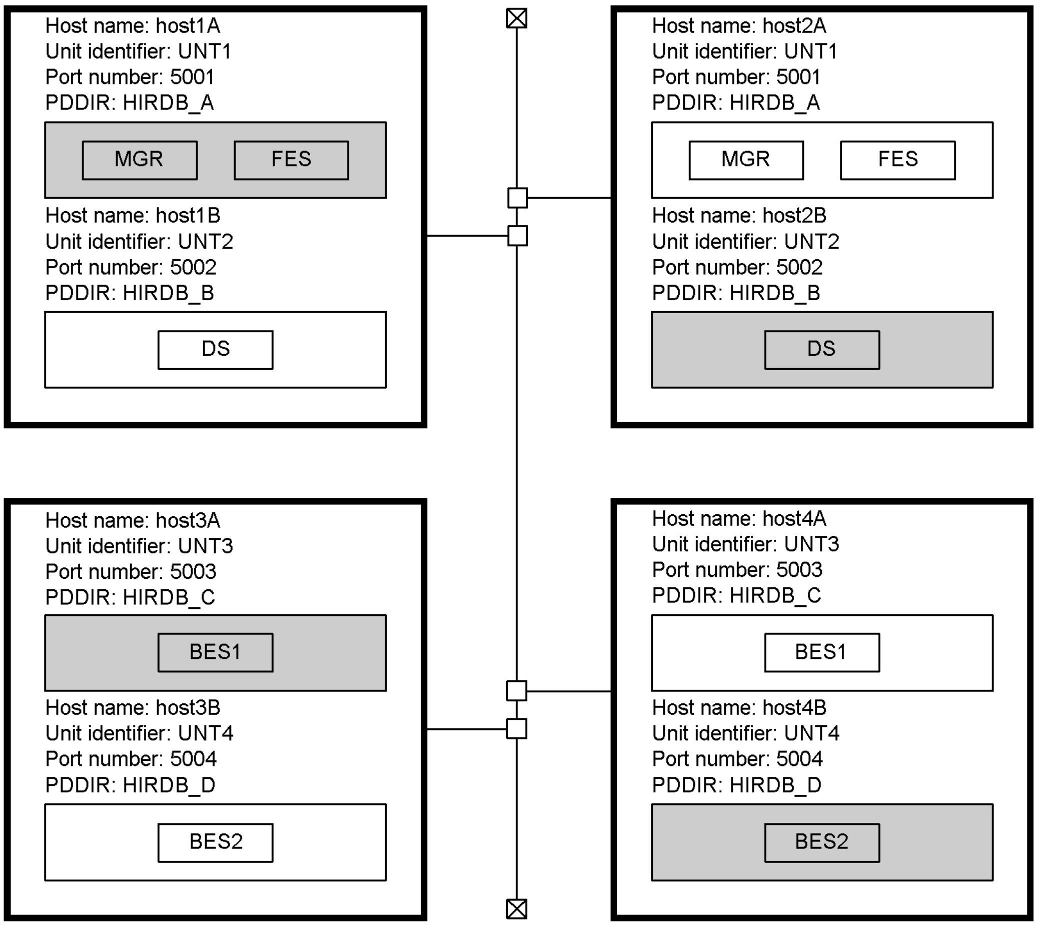
1. Specifies the action to be taken by the UAP when the RDAREA or index to be accessed using a local buffer is being used by another user.
2. Allocates a local buffer to a user RDAREA (`RDAREA10`).
3. Allocates a local buffer to user RDAREAs (`RDAREA11` and `RDAREA12`).
4. Allocates a local buffer to an index (`INDX01`).

B.4 HiRDB/Parallel Server: when the standby system switchover facility is used

This example creates system definitions for a HiRDB/Parallel Server. The system configuration is shown as follows.

- Mutual system switchover is used.
- IP addresses are not inherited.
- The rapid system switchover facility is used for all units.

Note that in this definition example, only those operands that are related to the system common definition and unit control information definition are explained.



Note: Shading indicates the primary HiRDB systems.

Key points

- When IP addresses are not inherited, specify a host name using an alias IP address for the `-x` and `-c` options of the `pdunit` operand, and not the standard host name, to maintain independence from the server machine.
- When mutual system switchover without IP address inheriting is used, the host name to be specified for the `-x` and `-c` options of the `pdunit` operand must be unique. See the `pdunit` operand specification example in (*1*).

(1) System common definition

```
         :
set pd_name_port = 5001                                        1
         :
set pd_ha = use                                                2
set pd_ha_ipaddr_inherit = N
set pd_ha_switch_timeout = Y
set pd_ha_transaction = queuing
set pd_ha_trn_queuing_wait_time = 240
set pd_ha_trn_restart_retry_time = 90
         :
pdunit -x host1A -u UNT1 -d "C:\HiRDB_A" -c host2A -p 5001     3
pdunit -x host2B -u UNT2 -d "C:\HiRDB_B" -c host1B -p 5002
pdunit -x host3A -u UNT3 -d "C:\HiRDB_C" -c host4A -p 5003
pdunit -x host4B -u UNT4 -d "C:\HiRDB_D" -c host3B -p 5004
pdstart -t MGR -u UNT1                                         4
pdstart -t FES -s fes1 -u UNT1
pdstart -t DIC -s dic  -u UNT2
pdstart -t BES -s bes1 -u UNT3
pdstart -t BES -s bes2 -u UNT4
         :
```

Explanation

1. Specifies the HiRDB port number.
2. Specifies the operands related to the system switchover facility.
3. Specifies the unit configuration of HiRDB/Parallel Server:
   `-x`: Host name
   `-u`: Unit identifier
   `-d`: HiRDB directory name
   `-c`: Host name of the secondary system
   `-p`: Port number of the unit
4. Specifies the server configuration of the HiRDB/Parallel Server:
   `-t`: Server type
   `-s`: Server name
   `-u`: Unit identifier

(2) Unit control information definitions for UNT1-UNT4

```
         :
set pd_ha_acttype = server                     1
set pd_ha_agent = standbyunit                  2
         :
```

Explanation

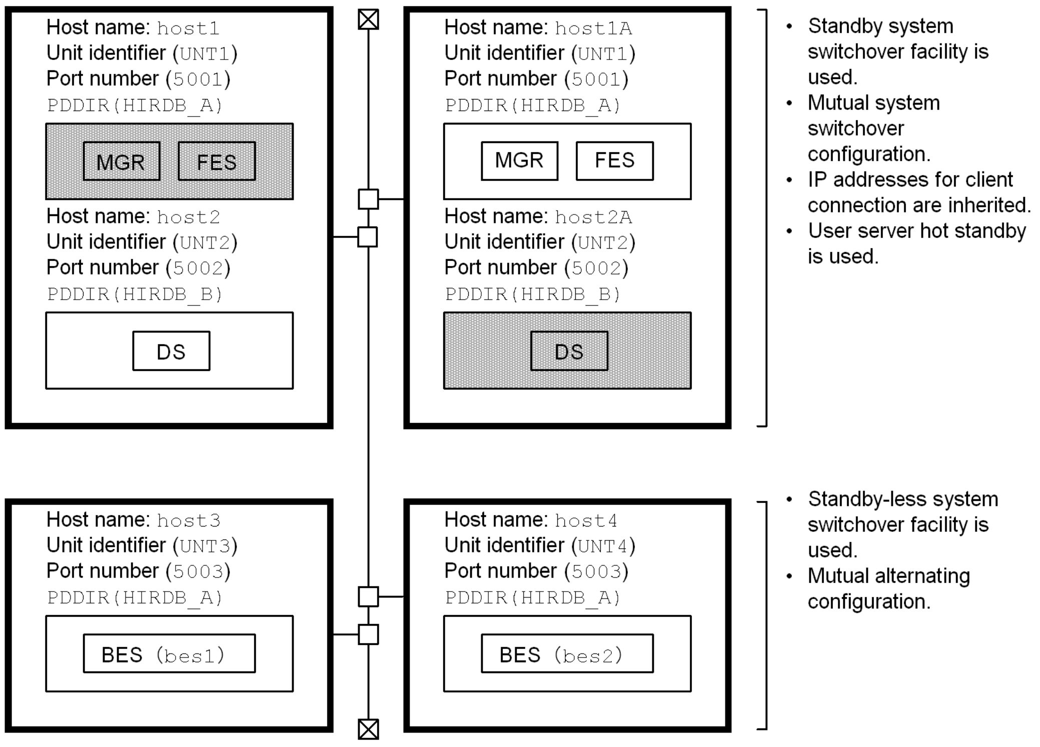
1. Specifies that the system switchover facility be used in the server mode.
2. Specifies that the rapid system switchover facility be used.

B.5 HiRDB/Parallel Server: when the standby-less system switchover (1:1) facility is used

This example creates system definitions for a HiRDB/Parallel Server. The system configuration is shown as follows.

- Both standby and standby-less system switchover facilities are used.
- The units to which the standby system switchover facility is applied are based on a mutual system switchover configuration and inherit IP addresses for client connection only. User server hot standby is also used.
- The units to which the standby-less system switchover (1:1) facility is applied are based on a mutual alternating configuration.

Note that in this definition example, only those operands that are related to the system common definition and unit control information definition are explained.



Note

- Shading indicates the primary units.
- Server name is shown in parentheses.
- Back-end server (bes1) is a normal BES and is also an alternate BES for back-end server (bes2).
- Back-end server (bes2) is a normal BES and is also an alternate BES for back-end server (bes1).

(1) System common definition

```
         :
set pd_name_port = 5001                                       1
         :
set pd_ha = use                                               2
set pd_ha_ipaddr_inherit = N
set pd_ha_switch_timeout = Y
set pd_ha_transaction = queuing
set pd_ha_trn_queuing_wait_time = 240
set pd_ha_trn_restart_retry_time = 90
         :
pdunit -x host1 -u UNT1 -d "C:\HiRDB_A" -c host1A -p 5001        3
pdunit -x host2 -u UNT2 -d "C:\HiRDB_B" -c host2A -p 5002
pdunit -x host3 -u UNT3 -d "C:\HiRDB_A" -p 5003
pdunit -x host4 -u UNT4 -d "C:\HiRDB_A" -p 5003
pdstart -t MGR -u UNT1                                         4
pdstart -t FES -s fes1 -u UNT1
pdstart -t DIC -s dic  -u UNT2
pdstart -t BES -s bes1 -u UNT3 -c bes2                         5
pdstart -t BES -s bes2 -u UNT4 -c bes1
         :
```

Explanation

1. Specifies the HiRDB port number.
2. Specifies the operands related to the system switchover facility.
3. Specifies the unit configuration of HiRDB/Parallel Server:
   `-x`: Host name
   `-u`: Unit identifier
   `-d`: HiRDB directory name
   `-c`: Host name of the secondary system
   `-p`: Port number of the unit
4. Specifies the server configuration of the HiRDB/Parallel Server:
   `-t`: Server type
   `-s`: Server name
   `-t`: Server type
5. Specifies the server configuration of the HiRDB/Parallel Server:
   `-t`: Server type
   `-s`: Specifies the name of the normal BES.
   `-t`: Server type
   `-c`: Specifies the name of the alternate BES.

(2) Unit control information definitions for UNT1 and UNT2

```
         :
set pd_ha_acttype = server                                      1
set pd_ha_server_process_standby = Y                            2
         :
```

Explanation

1. Specifies that the system switchover facility be used in the server mode.
2. Specifies that user server hot standby be used.

(3) Unit control information definitions for UNT3 and UNT4

```
         :
set pd_ha_acttype = server                                      1
set pd_ha_agent = server                                        2
         :
```

Explanation

1. Specifies that the system switchover facility be used in the server mode.
2. Specifies that the standby-less system switchover (1:1) facility be used.

C. Formulas for Determining Operand Specification Values

This appendix provides and explains formulas for determining specification values for various system definition operands. This appendix contains the following sections:

1. Formulas for determining size of statistics log file (`pd_stj_file_size`)
2. Formulas for determining size of SQL object buffer (`pd_sql_object_cache_size`)
3. Formulas for determining size of view analysis information buffers (`pd_view_def_cache_size`)
4. Formulas for determining size of table definition information buffer (`pd_table_def_cache_size`)
5. Formulas for determining total number of tables and RDAREAs per server locked with `UNTIL DISCONNECT` specified (`pd_lck_until_disconnect_cnt`)
6. Formulas for determining size of routine definition information buffer (`pd_routine_def_cache_size`)

C.1 Formulas for determining size of statistics log file (pd_stj_file_size)

The size of the statistics log that is output differs depending on the statistical information. The size of the statistics log for each type of statistical information is described starting in (1) as follows. For the `pd_stj_file_size` operand, specify a value that is equal to or greater than the combined total of the statistics log sizes determined in (1) and beyond.

Note

Each of these formulas produces a size in bytes; each result in bytes must be converted subsequently to kilobytes.

(1) Statistical information related to system operation (sys)

This will depend on the options specified in the `pdstbegin` operand or `pdstbegin` command.

When neither -a nor -s option is specified

Statistics log size = 2,412 × (↓ *a* ÷ *b* ↓ + *number-times-pdstjsync-command-is-executed-during-statistical-information-output-time*) (bytes)

When -a option is specified

Statistics log size = (1,836 × *c* + 2,412) × (↓ *a* ÷ *b* ↓ + *number-times-pdstjsync-command-is-executed-during-statistical-information-output-time*) (bytes)

When -s option is specified

Statistics log size = (1,836 × *d* + 2,412) × (↓ *a* ÷ *b* ↓ + *number-times-pdstjsync-command-is-executed-during-statistical-information-output-time*) (bytes)

a:

Statistical information output period (minutes)

b:

Statistical information output interval (minutes)

Interval specified in the `pdstbegin` operand or the `-m` option of the `pdstbegin` command

c:

Number of servers in unit

System manager is not included.

d:

Number of servers specified in the `pdstbegin` operand or the `-s` option of the `pdstbegin` command

(2) Statistical information related to UAPs (uap)

Formula

Statistics log size = 1,252 × *number-of-UAPS-to-be-executed-during-statistical-information-collection* (bytes)

If the total number of UAPs that will actually be executed is known, use the formula above. If the number of UAPs that will be executed during statistical information collection is not known, that number can be determined with the following formula:

- *Number of UAPs* = *number-of-UAPS-to-be-executed-per-unit-time* × *statistical-information-collection-period* ÷ *unit-time*

For example, if 10 UAPs are executed in 30 minutes and statistical information is collected for a period of 60 minutes, the UAP count will be 10 × 60 ÷ 30 = 20.

(3) Statistical information related to SQL (sql)

Formula

Statistics log size = 728 × *number-of-SQLs-to-be-executed-during-statistical-information-collection*# (bytes)

If the total number of SQLs that will actually be executed is known, use the formula above. If the number of SQLs that will be executed during statistical information collection is not known, that number can be determined with the following formula:

Number of SQLs# = *average number of SQL statements executed by UAP* × (*number of UAPs to be executed per unit time* × *statistical information collection period* ÷ *unit time*)

#: If a stored procedure or stored function will be executed, the SQL statements in the procedure or function must also be counted.

(4) Statistical information related to global buffers (buf)

Formula

Statistics log size = 412 × *a* × *b* (bytes)

a:

Number specified by the `pdbuffer` operand

b:

Number of times synchronization point dump is collected during statistical information collection

The number of times a synchronization point dump is collected during the statistical information collection period can be determined with the following formula:

↑ *size-of-system-log-output-during-statistical-information-collection-period*

÷ (*total-of-pd_log_max_data_size-operand-values-in-individual-server-definitions* ÷ 3)

× *total-of-pd_log_sdinterval-operand-specification-values-in-individual-server-definitions* ↑

For details about the system log size, see the *HiRDB Version 9 Installation and Design Guide*.

(5) HiRDB file statistical information related to database manipulation (fil)

Formula

Statistics log size = 428 × *a* × *b* (bytes)

a:

Number of HiRDB files specified in the control statement of the database initialization utility

b:

Number of times synchronization point dump is collected during statistical information collection

The number of times a synchronization point dump is collected during the statistical information collection period can be determined with the following formula:

↑ *size-of-system-log-size-output-during-statistical-information-collection-period*

÷ (*total-of-pd_log_max_data_size-operand-specification-values-in-individual-server-definitions* ÷ 3)

× *total-of-pd_log_sdinterval-operand-specification-values-in-individual-server-definitions* ↑

For details about the system log size, see the *HiRDB Version 9 Installation and Design Guide*.

(6) Statistical information related to deferred write processing (dfw)

Formula

Statistics log size = $384 \times a \times 2 + (384 \times b \times c) \div (d \times e \times f)$ (bytes)

a:

Number of times synchronization point dump is collected during statistical information collection

The number of times a synchronization point dump is collected during the statistical information collection period can be determined with the following formula:

$\uparrow$ *size-of-system-log-size-output-during-statistical-information-collection-period*

$\div$ (*total-of-pd_log_max_data_size-operand-specification-values-in-individual-server-definitions* $\div$ 3)

$\times$ *total-of-pd_log_sdinterval-operand-specification-values-in-individual-server-definitions* $\uparrow$

For details about the system log size, see the *HiRDB Version 9 Installation and Design Guide*.

b:

Average number of pages updated per transaction

c:

Number of transactions to be executed within statistical information collection period

d:

Total buffer sectors count specified by the `-n` option of the `pdbuffer` operand

e:

Database-updating transactions as a percentage of all transactions

f:

Updated pages output ratio during deferred write, as specified by the `pdbuffer` operand

(7) Statistical information related to indexes (idx)

Formula

Statistics log size = $3{,}768 \times \uparrow a \div 128 \uparrow \times b$ (bytes)

a:

Total number of partitioned indexes

The total number of partitioned indexes is the sum of the numbers of RDAREA names specified by IN RDAREA of individual `CREATE INDEX` statements of definition SQLs.

b:

Number of times synchronization point dump is collected during statistical information collection

The number of times a synchronization point dump is collected during the statistical information collection period can be determined with the following formula:

$\uparrow$ *size-of-system-log-output-during-statistical-information-collection-period*

$\div$ (*total-of-pd_log_max_data_size-operand-specification-values-in-individual-server-definitions* $\div$ 3)

$\times$ *total-of-pd_log_sdinterval-operand-specification-values-in-individual-server-definitions* $\uparrow$

For details about the system log size, see the *HiRDB Version 9 Installation and Design Guide*.

(8) Statistical information related to SQL static optimization (sop)

Formula

Statistics log size = $92 \times a$ (bytes)

a:

Number of SQL objects to be created in the SQL statements to be executed during statistical information collection. This applies to the following SQL statements: `PREPARE` statement, `EXECUTE IMMEDIATE` statement, and static SQL statements.

If it is possible to determine the actual number of SQL objects to be created, this formula can be used to make that determination. If the number of SQL objects to be created during statistical information collection is not known, the number of SQL objects to be created can be determined using the following formula:

Number of SQL objects to be created = average-number-of-SQL-statements-to-be-executed-in-UAP × *SQL-object-cache-miss-rate* × *(number-of-UAP-executions-per-unit-of-time* × *statistical-information-collection-time* ÷ *unit-of-time)*

The SQL object cache miss rate can be determined from the statistical information related to the system.

(9) Statistical information related to SQL dynamic optimization (dop)

Formula

Statistics log size =

$$\sum_{i=1}^{a} \left\{ \begin{array}{l} 28 + 16 \times \textit{number-of-tables-inside-SQL} \times 2 \\ + 12 \times (1 + \sum_{j=1}^{b} \textit{number-of-RDAREAs-that-define-table-c} \times 2 \end{array} \right\}$$

(bytes)

a: Number of SQL statements to be executed

b: Number of tables inside the *c*-th SQL statement

(10) Statistical information related to SQL object execution (pcd)

- HiRDB/Single Server

Formula

Statistics log size =388 × $\{(a + b + c + d + e + f) + g + h\}$

(bytes)

- HiRDB/Parallel Server

Formula

Statistics log size =388 × $\{(a + b + c + d + e + f) \times i + g + h\}$

(bytes)

a: Number of `OPEN` statements to be executed

b: Number of `CLOSE` statements to be executed

c: Number of `INSERT` statements to be executed

d: Number of `DELETE` statements to be executed

e: Number of `ASSIGN LIST` statements to be executed

f: Number of `UPDATE` statements to be executed

g: Number of `FETCH` statements to be executed

h: Number of `DESCRIBE` statements to be executed

i: Value obtained from the following formula:

$$\sum_{k=1}^{n} \left\{ 1 + \sum_{m=1}^{Tm} \textit{number-of-RDAREAs-that-define-table-m} \times 2 \right\}$$

n: Number of SQL statements to be executed

Tm: Number of tables inside the *m*-th SQL statement

(11) Statistical information related to SQL statement statistics (sqh)

Formula

Statistics log size = (728 + *a*) × *b* (bytes)

a: Average SQL length of SQL statements

b: Number of SQLs to be executed during statistical information collection

If the total number of SQLs that will actually be executed is known, use the computation formula above. If the number of SQLs that will be executed during statistical information collection is not known, that number can be determined with the following formula:

Number of SQLs$^{\#}$ = *average-number-of-SQL-statements-executed-per-UAP* $\times$ (*number-of-UAPs-to-be-executed-per-unit-time* $\times$ *(statistical-information-collection-period* $\div$ *unit-time*)

#: If a stored procedure or stored function will be executed, the SQL statements in the procedure or function must also be counted.

(12) Statistics on SQL object transmission (obj)

Formula

Statistics log size = 300 $\times$ *number-of-SQL-statements-to-be-executed-during-statistical-information-collection* (bytes)

If it is possible to determine the cumulative number of SQL statements to be executed, this formula can be used to make that determination. If the number of SQL statements to be executed during statistical information collection is not known, the number of SQL statements can be determined using the following formula:

Number of SQL statements$^{\#}$ = *average-number-of-SQL-statements-to-be-executed-in-UAP* $\times$ *(number-of-UAP-executions-per-unit-of-time* $\times$ *statistical-information-collection-time* $\div$ *unit-of-time)*

#: If a stored procedure or stored function is to be executed, the SQL statements described in the procedure or function must also be included in the count.

C.2 Formulas for determining size of SQL object buffer (pd_sql_object_cache_size)

For information on the variables used in the formulas, see (3) following.

(1) Formula for determining the size of the SQL object in an SQL statement

The following formula is used to determine the size of the SQL object in an SQL statement:

Formula

Size of the SQL object in an SQL statement =

$\uparrow\{$

$1{,}600 + 46 \times RCN + 394 \times Si + 24 \times Pi + 2{,}058 \times Ti + 76 \times Ti \times \times Di + 80 \times Ci + 40 \times Ii + 586 \times Wi$

$+ 24 \times Ki + Li + 8 \times TCi + 656 \times Di + 116 \times Ti \times QX + 28 \times QX + 200 \times Ai + 48 \times nFF$

$+ 100 \times nFP + 148 \times nFC + 712 \times nPFF + 32 \times (nAT + nPAT) + 20 \times nCAT$

$+ 28 \times (nAF + nCAF) + 20 \times (nAA + nPAA + nCAA) + 1{,}057 \times nSPA + 120 \times nSPP$

$+ 287 \times nSFF + 8 \times nSFP + 813 \times nJFC + 20 \times nJFP$

Add the result of this formula when using a trigger:

$+ 1{,}057 \times nTR + 120 \times (nTSN + nTSO) + 20 \times (nTCN + nTCO)$

Add the result of this formula when using a referential constraint:

$+ 760 + 376 \times RCC + 1{,}880 \times RCT$

Add the result of this formula when using a set operation:

$+ 64 \times Ui$

Add the result of this formula when using matrix partitioning:

$+ 16 \times Tmi + 16 \times Tmi \times Di$

Add the result of this formula when specifying a row value constructor:

$+ 384 \times Wri + 32 \times QXs$

Add the result of this formula when using a check constraint:

$+ 72 \times CDi + 88$

Add the result of this formula when obtaining SQL runtime interim results:

$+ 24 \times PIX + 192 \times Ti + 68 \times QX + 96$

Add the result of this formula when specifying the `LIMIT` clause:

$+ 160$

Add the result of this formula when defining a falsification prevented table:

$+ 200$

Add the result of this formula when using the `SET SESSION AUTHORIZATION` statement:

$+ 32$

Add the result of this formula when obtaining transfer values from a list:

$+ 32 \times ALP$

Add the result of this formula when using the `XML` type:

$+ 36 \times Ti + 4{,}240 \times XQX + 8 \times XQX \times nURI$

Add the result of this formula when using a character set:

$+ 80 \times TCCi + 88 \times sRi$

Add the result of this formula when using a sequence generator:

$+ 168 \times nSQ$

Add the result of this formula when using the data compression facility:

$+ 8 \times Ci + 32 \times nUOC + 28$

Add the result of this formula when using temporary tables:

$+ 200 \times nTT$

This is a computation formula for column name description area size. Add the result of this formula when using dynamic SQL:

$+ 32 \times Si + 16$

This is a computation formula for type name description area size. Add the result of this formula when using dynamic SQL:

$+ \uparrow (42 \times SiT) + \{52 + 152 \times (SiTA + SiSA + SiNA) \times (SiT + SiS + SiN)\} \uparrow$

$\} \div 1{,}024 \uparrow$

Notes

- If you use stored procedures or stored functions, also include the SQL statements described in the procedures or functions in the computation.
- If you use triggers, also include the SQL statements described in the triggers in the computation.
- If you use stored procedures, stored functions, or triggers, compute the size of the routine control object separately for the stored procedures, stored functions, or triggers, and add the result to the size of the SQL object buffer. The formula for computing the size of a routine control object is shown in (2) following.

(2) Formula for determining the size of the routine control object of a routine

(a) When the user defines a trigger

When the user defines a stored procedure, a stored function, or a trigger, the following formula is used to determine the size of the routine control object of a routine:

Formula

Size of the routine control object of a routine (in kilobytes) =

$\uparrow \{$

$600 + 28 \times sRi + 32 \times (sRUi + sDi) + 56 \times sSXi + sCUi + sSi + sPi + sLA + sKi + sL + 80 \times sWi + 24 \times sCM + 32 \times sCCR + 2 \times sDCR + 60 \times sCHD + 72 \times sDHD + 64 \times sHCN + 8 \times sCHD \times sHCN + 48 \times nRFF + 100 \times nRFP + 148 \times nRFC + 200 \times nPRFF + 8 \times nPRFP + 196 \times nPA + 64 \times nPP + 36 \times nPPI + 20 \times nPPO + 200 \times nPPA + 8 \times nPPP + 20 \times nAR + 48 \times nARA + 16 \times nRPAT + 20 \times nCAT + 28 \times (nRPAF + nRCAF) + 20 \times (nRPAA + nRCAA) + 287 \times nRSFF + 8 \times nRSFP + 813 \times nPJA + 20 \times nPJP + 813 \times nRJFC + 20 \times nRJFP$

Add the result of this formula when using a trigger:

$+ 28 \times (nTSN \times 2 + nTSO)$

$\} \div 1{,}024 \uparrow$

(b) When HiRDB automatically creates a trigger

When HiRDB creates a trigger for constraint control when `CASCADE` is specified for the referential action during table definition, the following formula is used to determine the size of the routine control object of a routine:

Formula

Size of the routine control object of a routine (in kilobytes) =

$$\uparrow \{ 608 \times RCC + (5{,}120 + 100 \times RDi + 256 \times RIi) \times RCP \times RCT \} \div 1{,}024 \uparrow$$

(3) Variables used in the formulas

Variable name	Explanation
RCN	Total number of tables and indexes used by the SQL object
Si	Number of retrieval items inside the SQL statement (number of columns if the columns specified by the SQL statement are index columns)
Pi	Number of embedded variables or parameters inside the SQL statement
Ti	Number of table names inside the SQL statement
Ci	Number of column names inside the SQL statement Also add the number of columns for which the following definition was specified during table definition (these columns might not be in the SQL statement): • Columns for which `DEFAULT` is specified in the column definition • Columns for which `WITH DEFAULT` is specified for a `NOT NULL` constraint specification
TCi	Number of table columns inside the SQL statement
Wi	Number of boolean operators inside the SQL statement#
Ki	Number of constants inside the SQL statement#
Li	Total size of the constants inside the SQL statement# (in bytes)
Ii	Number of indexes used during SQL statement execution (number of indexes specified as retrieval conditions in the table specified by the SQL statement)
Di	Total number of storage conditions defined in the table inside the SQL statement (times 2 for a matrix-partitioned table)
QX	Number of query specifications
Ai	Total number of scalar operations in the SQL statement
SiT	Number of abstract data types in the selection expression inside the SQL statement
SiS	Number of supertypes for the abstract data types in the selection expression inside the SQL statement
SiN	Total number of supertypes for the abstract data type, which is a subtype of the selection expression inside the SQL statement
SiTA	Number of attributes for the abstract data types in the selection expression inside the SQL statement
SiSA	Number of attributes of the supertypes for the abstract data types in the selection expression inside the SQL statement
SiNA	Total number of components specified for the abstract data type, which is a subtype of the selection expression inside the SQL statement
nSPA	Number of times the procedure statements inside the SQL statement are invoked
nSPP	Total number of arguments for the procedure statements inside the SQL statement
nFF	Number of times the functions inside the SQL statement are invoked#
nFP	Total number of function arguments inside the SQL statement#

Variable name	Explanation
nFC	Total number of function definition candidates for the functions inside the SQL statement (number of function invocations *nFF* + number of function definitions that use as arguments the subtypes to the function invocations whose arguments are abstract data types)
nPFF	Number of times the plug-in functions used by SQL objects are invoked (1 for the number of plug-in function invocations inside the SQL statement + `SELECT`; 6 for `INSERT`, `UPDATE`, or `DELETE`)
nSFF	Number of times the system definition scalar functions inside the SQL statement are invoked#
nSFP	Total number of arguments for the system definition scalar functions inside the SQL statement#
nJFC	Number of times external Java functions are invoked inside the SQL statement
nJFP	Total number of arguments of external Java functions inside the SQL statement
nAT	Number of abstract data types used by component specification inside the SQL statement (excluding the abstract data types that are manifested by supertypes and abstract data type attributes)
nAA	Number of abstract data types used by component specification inside the SQL statement (including the abstract data types that are manifested by supertypes and abstract data type attributes)
nAF	Total number of attributes used by component specification inside the SQL statement
nPAT	Number of abstract data types used by the arguments of the plug-in functions used by the SQL object (excluding the abstract data types that are manifested by supertypes and abstract data type attributes)
nPAA	Number of abstract data types used by the arguments of the plug-in functions used by the SQL object (including supertypes and subtypes)
nCAT	Number of times the constructor functions inside the SQL statement are invoked
nCAA	Number of abstract data types of the constructor functions inside the SQL statement (including supertypes)
nCAF	Total number of attributes of the abstract data types of the constructor functions inside the SQL statement
nTR	Number of triggers that are activated by SQL statement execution
nTSN	Total number of columns modified by the new values correlation name inside the trigger SQL statement of each trigger activated by SQL statement execution
nTSO	Total number of columns modified by the old correlation name inside the trigger SQL statement of each trigger activated by SQL statement execution
nTCN	Total number of columns modified by the new values correlation name inside the trigger action condition of each trigger activated by SQL statement execution
nTCO	Total number of columns modified by the old correlation name inside the trigger action condition of each trigger activated by SQL statement execution
RCC	Total number of member columns of the foreign key and the primary key of the table that references the update-target table inside the SQL statement
RCT	Total number of tables that reference the update-target table and the number of tables that the update-target table references inside the SQL statement
RCP	Total number of cascades specified for referencing actions during referencing table definition
RIi	Total number of indexes defined for referenced tables that are specified for referencing during referencing table definition
RDi	Total number of partitioning storage conditions defined for referenced tables that are specified for referencing during referencing table definition (to be multiplied by 2 for a matrix partitioning table)
Ui	*Number of set operations* + 1
Tmi	Number of tables with matrix partitioning
Wri	Number of conditions using a row value constructor with two or more row value constructor elements specified

Variable name	Explanation
QXs	Number of table subqueries with multiple selection expressions specified
CDi	Number of check constraint conditions
PIX	Number of indexes used as multiple indexes
ALP	Number of receive functions for passing inter-function values from a list
XQX	Number of `XQuery` specifications
nURI	Number of XML name space URIs specified
TCCi	Number of columns with character set specification
nSQ	Number of sequence generators
nUOC	Number of UOCs used for data compression
nTT	Number of temporary tables
sRi	Number of SQL parameters inside procedures and functions (multiply the number of SQL parameters with the `INOUT` specification by 2).
sRUi	Total number of SQL parameters inside procedures and functions (or the total number of columns modified by an old or new values correlation name inside the trigger SQL statement inside trigger definition)
sDi	Number of SQL variables (`declare`) inside procedures, functions, and trigger SQL statements
sSXi	Total number of SQLCODE and SQLCOUNT variables inside procedures, functions, and trigger SQL statements
sCUi	Total number of `CURRENT_TIME` and `CURRENT_DATE` constants inside procedures, functions, and trigger SQL statements
sSi	Number of data manipulation SQL statements inside procedures and trigger SQL statements (SQL statements excluding cursor declaration: `OPEN`, `FETCH`, `CLOSE`, `UPDATE`, `DELETE`, and `INSERT` statements, for example)
sPi	Number of routine control SQL statements inside procedures, functions, and trigger SQL statements (`BEGIN`, `SET`, `IF`, `ELSE IF`, and `WHILE`, for example)
sLA	Number of labels inside procedures, functions, and trigger SQL statements
sKi	Number of constants inside procedures, functions, and trigger SQL statements (excluding the constants of data manipulation SQL statements described inside procedures and trigger SQL statements)
sL	Combined total size of the constants inside procedures, functions, and trigger SQL statements (excluding the constants of data manipulation SQL statements described inside procedures and trigger SQL statements)
sWi	Number of condition predicates inside procedures, functions, and trigger SQL statements
sCM	Number of compound statements inside procedures, functions, and trigger SQL statements
sCCR	Number of compound statements that describe cursor declarations for procedures and trigger SQL statements
sDCR	Number of cursor declarations for procedures and trigger SQL statements
sCHD	Number of compound statements that describe handler declarations for procedures, functions, and trigger SQL statements
sDHD	Number of handler declarations for procedures, functions, and trigger SQL statements
sHCN	Number of condition values described inside handler declarations for procedures, functions, and trigger SQL statements
nRFF	Number of function invocations inside the routine
nRFP	Total number of function arguments inside the routine
nRFC	Total number of function definition candidates for the functions inside the routine (number of function invocations *nFF* + number of function definitions that use as arguments the subtypes to the function invocations whose arguments are abstract data types)

Variable name	Explanation
nPRFF	Number of times the plug-in functions used by SQL objects of the routine are invoked
nPRFP	Total number of plug-in parameters of the plug-in functions used by the SQL objects of the routine
nPA	Number of procedure invocations inside the routine
nPP	Total number of parameters for the procedures inside the routine
nPPI	Total number of input parameters for the procedures inside the routine (including input/output parameters)
nPPO	Total number of output parameters for the procedures inside the routine (including input/output parameters)
nPPA	Number of times the plug-in procedure used by the SQL objects of the routine is invoked
nPPP	Total number of plug-in parameters of the plug-in procedure used by the SQL objects of the routine
nRSFF	Number of times the system definition scalar functions inside the routine are invoked
nRSFP	Total number of arguments for the system definition scalar functions inside the routine
nPJA	Number of times external Java procedures are invoked inside the routine
nPJP	Total number of arguments for external Java procedures inside the routine
nRJFC	Number of times external Java functions are invoked inside the routine
nRJFP	Total number of arguments for external Java functions inside the routine
nAR	Number of abstract data types used by component specification inside the routine (excluding the abstract data types that are manifested by supertypes and abstract data type attributes)
nARA	Total number of attributes used in component specification inside the routine
nRPAT	Number of abstract data types used by the parameters of the plug-in routines used by the SQL objects of the routine (excluding the abstract data types of supertypes and abstract data type attributes)
nRPAA	Number of abstract data types used by the parameters of the plug-in routines used by the SQL objects of the routine (including supertypes)
nRPAF	Total number of attributes of the abstract data types used by the parameters of the plug-in routines used by the SQL objects of the routine
nRCAT	Number of times the constructor functions inside the routine are invoked
nRCAA	Number of abstract data types of the constructor functions inside the routine (including supertypes)
nRCAF	Total number of attributes of the abstract data types of the constructor functions inside the routine

#: When triggers are used, the trigger action conditions for the individual triggers that are activated by SQL statement execution must also be counted.

C.3 Formulas for determining size of view analysis information buffers (pd_view_def_cache_size)

(1) Formulas for determining the size of the view analysis information buffer for a single view table

The size of the view analysis information buffer for a single view table can be found from the following formulas. See *(2)* and *(3)* for details about the variables used in the formulas.

Formula for HiRDB in 32-bit mode

Size of the view analysis information buffer per view table (in kilobytes) =
$\uparrow$ *view-analysis-information-size-per-view-table* ÷ 1,024 $\uparrow$

view-analysis-information-size-per-view-table (bytes) =

$\uparrow (1{,}024 + LCNST + 15) \div 16 \uparrow \times 16$
$+ \uparrow (LPTREE + 15) \div 16 \uparrow \times 16$
$+ \uparrow (28 \times NINCC + 15) \div 16 \uparrow \times 16$
$+ \uparrow (28 \times NINCP + 15) \div 16 \uparrow \times 16$
$+ \uparrow (24 + 512 \times NTBL + 40 \times (NTBL + NDTBL)$
$+ 128 \times NCLM + 15) \div 16 \uparrow \times 16$
$+ \uparrow (16 \times \mathrm{MAX}(\uparrow NADTL \div 50 \uparrow, 1)$
$+ 200 \times NADTL) \div 16 \uparrow \times 16$
$+ \uparrow (16 \times \mathrm{MAX}(\uparrow NATTL \div 50 \uparrow, 1)$
$+ 144 \times NATTL) \div 16 \uparrow \times 16$
$+ 16 + 16 \times (3 + NINCC + NINCP)$

Formula for HiRDB in 64-bit mode

Size of view analysis information buffer per view table (in kilobytes) =

$\uparrow$ *view-analysis-information-size-per-view-table* $\div$ 1,024 $\uparrow$

view-analysis-information-size-per-view-table (bytes) =

$\uparrow (1{,}600 + LCNST + 15) \div 16 \uparrow \times 16$
$+ \uparrow (LPTREE + 15) \div 16 \uparrow \times 16$
$+ \uparrow (40 \times NINCC + 15) \div 16 \uparrow \times 16$
$+ \uparrow (40 \times NINCP + 15) \div 16 \uparrow \times 16$
$+ \uparrow (32 + 768 \times NTBL + 48 \times (NTBL + NDTBL)$
$+ 184 \times NCLM + 15) \div 16 \uparrow \times 16$
$+ \uparrow (20 \times \mathrm{MAX}(\uparrow NADTL \div 50 \uparrow, 1)$
$+ 224 \times NADTL) \div 16 \uparrow \times 16$
$+ \uparrow (20 \times \mathrm{MAX}(\uparrow NATTL \div 50 \uparrow, 1)$
$+ 160 \times NATTL) \div 16 \uparrow \times 16$
$+ 16 + 24 \times (3 + NINCC + NINCP)$

Note

When a view definition includes a view table that generates an internal derived table, add the formula for the size of the view analysis information buffer for that table.

If the view table does not generate an internal derived table, add the value calculated for the SQL statement that is specified directly without going through its view table.

(2) LPTREE formulas

Determine *LPTREE* from the following formulas. See *(3)* for details about the variables used in the formulas.

Formula for HiRDB in 32-bit mode

LPTREE (bytes) =

$276 \times NQRY$
$+ 12 \times (2 \times NQRY + NSBQ + NSTOP)$
$+ 12 \times (2 \times NQRY + NSLST + NSLAS)$
$+ 12 \times (4 \times NTBL + 4 \times NJTBL + 5 \times NDTBL + NDCLM)$
$+ 12 \times (NWHRC + \mathrm{NGPHV})$
$+ 12 \times (NVLCM + NEXCM + NNLCM$
$+ 3 \times NRANG + 3 \times NLKCM + 3 \times NSMCM + NETCM)$
$+ 12 \times NRVCL$
$+ 12 \times NTVCL$
$+ 12 \times NLGEX$
$+ 12 \times (NCNST + NSREG + NPRCS)$
$+ 12 \times 4 \times NCSCV$

$+ 12 \times (NARTH + NCNCT)$
$+ 12 \times NSFNC$
$+ 12 \times NLBLD$
$+ 12 \times (3 \times NCLM)$
$+ 12 \times (\mathrm{NCASE} + \mathrm{NWHEN})$
$+ 12 \times 2 \times (NSCLF + 2 \times NCSSP)$
$+ 12 \times (2 \times NPOS + 3 \times NDTV)$
$+ 12 \times (NEXTR + 2 \times NSBST + 3 \times NVALU + NBTEX + NCAST)$
$+ 12 \times (4 \times NFCSP + NFPRM)$
$+ 12 \times (3 \times NATTL + NATNM)$
$+ 28 \times NOPTL + 20 \times NOPTJ + 32 \times NOPTT + 20 \times NOPTIX$
$+ 8 \times NSLAS$
$+ 120 \times NJTBL$
$+ 480 \times NCSET$
$+ 84 \times NSPDT$
$+ 72 \times (NSCLF + NCASE + NCSSP \times 2)$
$+ (144 + 20 + 32) \times NFCSP + (300 + 136) \times NRTNL$
$+ 260 \times (NEXRTN + NCLASS + NJAR) + 32 \times NPVOW$
$+ 140 \times NPLGL + 172 \times NPPRL$
$+ (260 + 257 + 257) \times NPLGL$
$+ 32 \times NCUD$
$+ 512 \times NDTBL$
$+ 40 \times NDTBL$
$+ 128 \times NDCLM$
$+ 182$
$+ 376$
$+ 1{,}384$
$+ 356 \times NVCLM$
$+ 24 \times NVCSC$

Formula for HiRDB in 64-bit mode

LPTREE (bytes) =
$496 \times NQRY$
$+ 24 \times (2 \times NQRY + \mathrm{NSBQ} + NSTOP)$
$+ 24 \times (2 \times NQRY + NSLST + NSLAS)$
$+ 24 \times (4 \times NTBL + 4 \times NJTBL + 5 \times NDTBL + NDCLM)$
$+ 24 \times (NWHRC + NGPHV)$
$+ 24 \times (NVLCM + NEXCM + NNLCM$
$+ 3 \times NRANG + 3 \times NLKCM + 3 \times NSMCM + NETCM)$
$+ 24 \times NRVCL$
$+ 24 \times NTVCL$
$+ 24 \times NLGEX$
$+ 24 \times (NCNST + NSREG + NPRCS)$
$+ 24 \times 4 \times NCSCV$
$+ 24 \times (NARTH + NCNCT)$
$+ 24 \times NSFNC$
$+ 24 \times NLBLD$
$+ 24 \times (3 \times NCLM)$
$+ 24 \times (NCASE + NWHEN)$
$+ 24 \times 2 \times (NSCLF + 2 \times NCSSP)$
$+ 24 \times (2 \times NPOS + 3 \times NDTV)$
$+ 24 \times (NEXTR + 2 \times NSBST + 3 \times NVALU + NBTEX + NCAST)$
$+ 24 \times (4 \times NFCSP + NFPRM)$
$+ 24 \times (3 \times NATTL + NATNM)$
$+ 56 \times NOPTL + 40 \times NOPTJ + 64 \times NOPTT + 40 \times NOPTIX$
$+ 16 \times NSLAS$

$+ 208 \times NJTBL$
$+ 480 \times NCSET$
$+ 96 \times NSPDT$
$+ 96 \times (NSCLF + NCASE + NCSSP \times 2)$
$+ (176 + 24 + 40) \times NFCSP + (328 + 144) \times NRTNL$
$+ 260 \times (NEXRTN + NCLASS + \mathrm{NJAR}) + 32 \times NPVOW$
$+ 168 \times NPLGL + 184 \times NPPRL$
$+ (260 + 264 + 257) \times NPLGL$
$+ 40 \times NCUD$
$+ 768 \times NDTBL$
$+ 48 \times NDTBL$
$+ 184 \times NDCLM$
$+ 226$
$+ 568$
$+ 1{,}496$
$+ 480 \times NVCLM$
$+ 32 \times NVCSC$

(3) Variables used in the formulas

Variable name	Explanation
LCNST	Total size of the constants resulting from adding the following expressions:#1, #2, #3 • Total size of constants in the view definition + number of character set conversions in the view definition • Total size after conversion of the target constants + number of numeric character conversions in the view definition • Total size after conversion of the target constants + number of scalar operations in the view definition
NINCC	• When *LCNST* > 11,516 bytes: 32-bit mode: $\uparrow (LCNST - 11{,}516) \div 4{,}020 \uparrow$ 64-bit mode: $\uparrow (LCNST - 11{,}516) \div 4{,}024 \uparrow$ • When *LCNST* $\leq$ 11,516 bytes: 0
NINCP	• When *LTREE* > 11,516 bytes: 32-bit mode: $\uparrow (LTREE - 11{,}516) \div 4{,}020 \uparrow$ 64-bit mode: $\uparrow (LTREE - 11{,}516) \div 4{,}024 \uparrow$ • When *LCNST* $\leq$ 11,516 bytes: 0
NQRY	Number of queries in the view definition
NSBQ	Number of subqueries in the view definition
NSTOP	Number of set operations in the view definition
NSLST	Total selection items for all query specifications in the view definition#4
NWHRC	Number of `WHERE` clauses in the view definition
NGPHV	Number of `GROUP BY` clauses in the view definition + Number of `HAVING` clauses in the view definition
NVLCM	Number of comparison predicates in the view definition + Number of quantified predicates in the view definition
NEXCM	Number of `IN` predicates in the view definition
NNLCM	Number of `NULL` predicates in the view definition
NRANG	Number of `BETWEEN` predicates in the view definition

Variable name	Explanation
NLKCM	Number of `LIKE` and `XLIKE` predicates in the view definition
NSMCM	Number of `SIMILAR` predicates in the view definition
NETCM	Number of `EXISTS` predicates in the view definition
NRVCL	Number of row value constructor elements in the view definition
NTVCL	Number of row value constructor elements on the right side of `IN` predicates in the view definition
NLGEX	Number of logical operators in the view definition
NCNST	Total number of constants in the view definition + Total number of pattern characters in the view definition
NSREG	Total number of `USER`, `CURRENT DATE`, `CURRENT TIME`, and `CURRENT TIMESTAMP` specifications in the view definition
NPRCS	Number of `CURRENT TIMESTAMP` decimal seconds precision specifications in the view definition
NCSCV	Number of constants subject to character set conversion in the view definition#2
NARTH	Total number of arithmetic, date, and time operations in the view definition
NCNCT	Number of concatenation operations in the view definition
NLBLD	Number of labeled durations in the view definition
NSFNC	Number of set functions in the view definition
NSCLF	Number of system built-in scalar functions in the view definition
NFCSP	Number of function invocations or scalar functions in the view definition
NCLM	Total number of columns in the view definition
NCASE	Number of `CASE` expressions in the view definition + Number of `NULLIF CASE` abbreviations in the view definition
NWHEN	Number of `WHEN CASE` expressions in the view definition + 1
NCSSP	Number of `CAST` specifications in the view definition
NPOS	Number of system built-in `POSITION` scalar functions in the view definition
NDTV	Number of system built-in scalar functions (`DATE`, `TIME`, `TIMESTAMP`, or `VARCHAR_FORMAT`) in the view definition
NEXTR	Number of system built-in scalar functions (`YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, or `SECOND`) in the view definition
NSBST	Number of system built-in `SUBSTR` scalar functions in the view definition
NVALU	Number of system built-in `VALUE` scalar functions in the view definition + Number of `COALESCE CASE` abbreviations in the view definition
NBTEX	Number of system built-in `BIT_AND_TEST` scalar functions in the view definition
NCAST	Number of system built-in scalar functions (`INTEGER` or `DECIMAL`) in the view definition
NOPTL	Number of SQL optimization specifications in the view definition
NOPTJ	Number of joined format SQL optimization specifications in the view definition
NOPTT	Number of table SQL optimization specifications in the view definition
NOPTIX	Number of index SQL optimization specifications in the view definition
NSLAS	Number of select expression `AS` clauses in the view definition
NJTBL	Number of joined tables in the view definition

Variable name	Explanation
NCSET	Number of types of character sets that can be specified by HiRDB + 1[#5]
NSPDT	Number of `AS` *data-type* specifications in the view definition
NFPRM	Total number of function invocation arguments and system-defined scalar function arguments in the view definition
NRTNL	Total number of functions potentially invoked by function invocations and functions potentially invoked by system-defined scalar functions in the view definition
NEXRTN	Total number of external routines among functions that could potentially be invoked in the view definition
NCLASS	Total number of JAVA function classes in the view definition
NJAR	Total number of JAVA function JAVA archives in the view definition
NPVOW	Total number of public view owner names in the view definition
NDTBL	Number of `FROM` clause derived tables in the view definition + Number of view tables that generate internal derived tables in the view definition
NTBL	• When the total number of tables in the view definition is 15 or fewer and the total of the number of correlation names in the view definition and the NDTBL value is 15 or less: 15 • When the total number of tables in the view definition is 16 or more and the total of the number of correlation names in the view definition and the NDTBL value is 16 or more: Max(*Total number of tables in the view definition*, *Number of correlation names in the view definition* + *NDTBL value*)
NDCLM	Total number of columns of `FROM` clause derived tables in the view definition + Total number of columns of view tables that generate internal derived tables in the view definition
NVCLM	Number of columns derived by the view definition
NVCSC	Number of columns derived by the view definition that specify a character set other than the default character set
NADTL	Number of all abstract data types in the view definition, including those in inheritance relationships[#6]
NATTL	Number of component specifications in the view definition
NATNM	Number of attribute names of all component specifications in the view definition
NPLGL	Number of functions provided by plug-ins in the view definition
NPPRL	Number of parameters of functions provided by plug-ins in the view definition
NCUD	Number of abstract data type columns in the view definition

#1

Use the following to calculate the sizes of the various types of constants:

Constant type	Size (bytes)	Maximum size required by boundary adjustment (bytes)
Integer constant	4	7
Decimal constant	↑(*decimal constant precision* + 1) ÷ 2↑	↑(*decimal constant precision* + 1) ÷ 2↑ + 7
Floating-point numeric literal	8	15
Character string constant	2 + *character string constant character count*	2 + *character string constant character count* + 3
Hexadecimal constant	4 + (*hexadecimal constant character string length*) ÷ 2	4 + (*hexadecimal constant character string length*) ÷ 2 + 3
National character string literal	2 + *national character string literal character count* × 2	2 + *national character string literal character count* × 2 + 3

Constant type	Size (bytes)	Maximum size required by boundary adjustment (bytes)
Mixed character string literal	2 + *mixed character string literal single-byte character count* + *mixed character string literal double-byte character count* × 2	2 + *mixed character string literal single-byte character count* + *mixed character string literal double-byte character count* × 2 + 3

#2

The following constants are subject to character set conversion:

- Character string constants that are compared to character set items.
- Character string constants for which a character set was specified by a `CAST` specification in the post-conversion data type.

#3

The following constants are subject to numeric character conversion:

- Character string constants that are compared to numeric attribute items.
- Numeric constants specified in an operand of a concatenation operation.
- Character string constants specified in the operand of an arithmetic operation.

#4

When `*` is specified in a selection item, add the total column count for all tables specified in `FROM` clauses. When *table-name*`.*` is specified in a select expression, add the column count of that table.

#5

The only character set that can be specified in HiRDB is EBCDIK.

#6

For details about the inheritance relationships of abstract data types provided by plug-ins, see the applicable manuals for the specific plug-ins.

C.4 Formulas for determining size of table definition information buffer (pd_table_def_cache_size)

(1) Variables used in the formulas

a: Value of `DEFINITION_CACHE_SIZE` (in bytes) of a dictionary table (`SQL_TABLES` table)

If you do not know the value of `DEFINITION_CACHE_SIZE`, see *(3) Determining the table definition cache size*.

b: Number of columns in the table

c: Number of indexes in the table

d: Number of table-partitioning conditions

e: Number of RDAREAs for table

f: Number of RDAREAs for index

g: Number of BLOB columns

h: Number of abstract data types that include `BLOB` attributes

i: Total number of `BLOB` attributes in abstract data types

j: Number of abstract data types

k: Number of plug-in options

m: Total number of BLOB attributes in abstract data types

n: Number of index exceptional key values

p: Value of `STATISTICS_CACHE_SIZE` (in bytes) of a dictionary table (`SQL_TABLES` table)

Add this value when you use the `pdgetcst` command to obtain table optimization information. Because the result is in bytes, it must be converted into kilobytes before being entered in the formulas. If you do not know the value of `STATISTICS_CACHE_SIZE`, use the following formula to determine it:

$(2.6 \times q_1 + 3.0 \times q_2 + 0.04 \times c + 0.02) \times 1{,}024$ (bytes)

q_1: Total of the number of column optimization information items whose column data type is not `DECIMAL` and the number of column optimization information items that are `DECIMAL` type with a precision up to 31 digits.

q_2: Number of column optimization information items whose column data is `DECIMAL` type with a precision of 32 digits or more

r: Number of columns specified in the `DEFAULT` operand of `CREATE TABLE`

s: Total default size specified in the `DEFAULT` operand of `CREATE TABLE` (bytes)

Add the sizes of all columns for which default values are specified. If the default values might be increased, take the increases into consideration.

t: Number of authorization identifiers that have the same table identifier as the table name of the public view table

u: Number of triggers defined in the table

v: Number of columns used that are qualified by old or new value correlation names

w: Number of trigger action conditions

x: Total size of trigger action condition analysis tree (bytes)

For the formulas used to estimate the size of the trigger action condition analysis tree, see *Determining the size of a normal data dictionary RDAREA* in the *HiRDB Version 9 Installation and Design Guide*.

y: Number of trigger columns of the `UPDATE` trigger

z: Number of foreign keys defined in the table

aa: Number of foreign keys that reference the table's primary key

ab: Number of check constraints defined in the table

ac: Total size of analysis tree for check constraints (bytes)

For the formulas used to estimate the size of the analysis tree for check constraints, see *Determining the size of a normal data dictionary RDAREA* in the *HiRDB Version 9 Installation and Design Guide*.

ad: Number of index member substructure paths

ae: Total size of the substructure paths of that index

af: Total size of the analysis tree for the substructure paths of that index (bytes)

For the formulas used to estimate the size of the analysis tree for substructure paths, see *Determining the size of a normal data dictionary RDAREA* in the *HiRDB Version 9 Installation and Design Guide*.

ag: Number of compressed columns defined for the table

(2) Table definition information buffer size per table (in kilobytes)

To determine the table definition information buffer size per table, use the following approximation formulas. Note that for a view table, also determine the size of the base table or foreign table that becomes the base for the view table.

Table type	Formula for determining the table definition information buffer size per table (in kilobytes)
Table definition information buffer size per base table	$\downarrow (4 + \uparrow a \div 1{,}024 \uparrow + 0.01 \times b + \uparrow p \div 1{,}024 \uparrow + 7) \div 8 \downarrow \times 8$
Table definition information buffer size per view table	$\downarrow (4 + \uparrow a \div 1{,}024 \uparrow + 0.01 \times b + 7) \div 8 \downarrow \times 8$

(3) Determining the table definition cache size

(a) When you do not know the value of DEFINITION_CACHE_SIZE

If any of the following conditions is applicable, the value of `DEFINITION_CACHE_SIZE` cannot be determined, even if a dictionary table is retrieved:

- Table has not yet been defined
- Version was upgraded from HiRDB Version 4.0 (in this case, the value of `DEFINITION_CACHE_SIZE` is not correct)[#1]
- Table was converted from 32-bit-mode HiRDB to 64-bit-mode HiRDB[#1, #2]

#1: In this case, the value in `DEFINITION_CACHE_SIZE` is not the correct value.

#2: Whether a table has been converted from 32-bit-mode HiRDB to 64-bit-mode HiRDB can be determined by looking at the time of execution of the `pdvrup` command and the time of table creation. If the table's creation time is earlier, the table has been converted from the 32-bit mode to the 64-bit mode. The `pdvrup` command's execution time can be determined from `CHANGE_TIME` of `MASTER.SQL_TABLES` in the SQL_TABLES dictionary table. A table's creation time can be determined from `CREATE_TIME` for the created table in the `SQL_TABLES` dictionary table. These columns can be retrieved with the following SQL statements:

- `pdvrup` command execution time

```
select CHANGE_TIME from MASTER.SQL_TABLES
  where TABLE_SCHEMA='MASTER'
  and TABLE_NAME='SQL_TABLES'
```

- Table creation time

```
select CREATE_TIME from MASTER.SQL_TABLES
  where TABLE_SCHEMA='authorization-identifier'
  and TABLE_NAME='table-name'
```

(b) Formula for determining the table definition cache size

Use the following formulas to determine the table definition cache size.

- Base table: (Formula 1 + Formula 2) $\times$ 1,024 (bytes)
- View table: (Formula 1 + Formula 3) $\times$ 1,024 (bytes)

Formula type	Formula
Formula 1	■ For 32-bit mode HiRDB $0.6 + 0.13 \times b + 0.15 \times c + 0.18 \times (d + 1) + 0.01 \times (e + f)$ ■ For 62-bit mode HiRDB $0.9 + 0.19 \times b + 0.18 \times c + 0.18 \times (d + 1) + 0.01 \times (e + f)$
Formula 2	■ Add the following if a LOB column is defined for the table: $(0.02 + 0.01 \times e) \times g$ ■ Add the following if an abstract data type column that includes the `BLOB` attribute is defined for the table: $0.02 \times h + (0.02 + 0.01 \times e) \times i$ ■ Add the following if an abstract data type column is defined for the table: $0.3 \times j + 0.3 \times k$ ■ Add the following if an index exclusion key value is defined for the table: $0.3 \times n$ ■ Add the following if `SEGMENT REUSE` is specified for the table (excludes `SEGMENT REUSE NO`): $0.01 \times e$ ■ Add the following if a default value is specified for the `DEFAULT` operand in the table: $0.01 \times r + \uparrow s \div 1{,}024 \uparrow$

Formula type	Formula
	■ Add the following if a trigger is defined for the table: $0.3 \times u + 0.2 \times v + 0.1 \times w + \uparrow x \div 1{,}024 \uparrow + 0.2 \times y$ ■ Add the following if a foreign key is defined for the table: $0.3 \times z$ ■ Add the following for a referenced table: $0.1 + 0.1 \times aa$ ■ Add the following if a check constraint is defined for the table: $0.1 \times ab + \uparrow ac \div 1{,}024 \uparrow$ ■ Add the following for each substructure index (if substructure indexes are defined for the table): $0.1 \times ad + \uparrow ae \div 1{,}024 \uparrow + \uparrow af \div 1{,}024 \uparrow$ ■ Add the following for a compressed table: $0.08 \times ag$
Formula 3	■ Add the following for a public view table: $0.1 + (0.1 \times t)$ ■ Add the following if a view table member column is an abstract data type: $0.3 \times j$

C.5 Formula for determining total number of tables and RDAREAs per server locked with UNTIL DISCONNECT specified (pd_lck_until_disconnect_cnt)

The following formula is used to determine the total number of tables and RDAREAs per server locked with `UNTIL DISCONNECT` specified (`pd_lck_until_disconnect_cnt`). Note that the formula differs depending on the server type. The estimation formulas for the server common definition are the same as for back-end servers.

Formula

Server type	Formula
Single server	$\{a + b + o + p + q + (t \times 3) + (w \times 2) + u + v + x\} \times 2 + \{2 \times (f + g + i + 1) + (g + y + z)\} \times e$
Dictionary server	$(a + b) \times 2 + \{2 \times (f + g + i + 1) + (g + y + z)\} \times e$
Back-end server	$\{a + b + o + p + q + r + s + (t \times 3) + (w \times 2) + u + v + x\} \times 2 + \{2 \times (f + g + i + 1) + (g + y + z)\} \times e$

a:

Number of tables specified in `LOCK TABLE` statements with `UNTIL DISCONNECT` specified that are to be concurrently executed

b:

Number of RDAREAs that store the tables specified in `LOCK TABLE` statements with `UNTIL DISCONNECT` specified that are to be concurrently executed

e:

Number of utilities that are concurrently executed

f:

Number of LOB RDAREAs that store the tables to be processed by the utilities that are concurrently executed

g:

Number of RDAREAs that store the indexes to be processed by the utilities that are concurrently executed *i*:

Number of RDAREAs that store the tables to be processed by the utilities that are concurrently executed.

o:

Number of RDAREAs that store the tables to which the local buffers for data that are to be used concurrently are allocated

Add this number when using local buffers for data.

p:

Number of RDAREAs that store the tables to which the local buffers for index that are to be used concurrently are allocated

Add this number when using local buffers for index.

q:

Total number of tables that are the target of the indexes to which the local buffers for index that are to be used concurrently are allocated

Add this number when using local buffers for index.

r:

Total number of shared table storage RDAREAs specified by `LOCK` statements with `IN EXCLUSIVE MODE` specified that are to be concurrently executed

Add this number when using shared RDAREAs.

s:

Total number of all shared RDAREAs for index defined in the shared tables that are specified by `LOCK` statements with `IN EXCLUSIVE MODE` specified that are to be concurrently executed

Add this number when using shared RDAREAs.

t:

Number of SQL session-specific temporary tables that exist concurrently

Add this number when using SQL session-specific temporary tables.

u:

Total number of temporary table RDAREAs for storing the SQL session-specific temporary tables that exist concurrently

Add this number when using SQL session-specific temporary tables.

v:

Total number of temporary table RDAREAs for storing the temporary table indexes defined in the SQL session-specific temporary table that exist concurrently

Add this number when using SQL session-specific temporary tables.

w:

Number of SQL session-specific temporary table indexes that exist concurrently

Add this number when using SQL session-specific temporary tables.

x:

Total number of segments used in the RDAREAs that store the SQL session-specific temporary tables and SQL session-specific temporary table indexes that exist concurrently

Add this number when using SQL session-specific temporary tables.

y:

Number of sequence generators that are used

Add this number when using the database load utility.

z:

Number of RDAREAs for storing the sequence generators that are used

Add this number when using the database load utility.

Note: Utilities in this case includes the following:

- Database load utility
- Database reorganization utility
- Free page release utility

- Global buffer residence utility
- Rebalancing utility (exclusive mode)

C.6 Formulas for determining size of routine definition information buffer (pd_routine_def_cache_size)

The following formula is used to determine the definition information size for routines:

total size of definition information for routines that are used frequently
+ total size of definition information for plug-in functions of plug-ins used
+ total size of definition information for system definition scalar functions that are used frequently

(1) Determination of routine definition information size per routine

The following formula is used to determine one routine's definition information size:

Formula

$\uparrow (1.3 + 0.2 \times a) \uparrow \times b$ (kilobytes)

a: Total number of parameters for routines that are used frequently

b: Number of definitions for routines that are used frequently

(2) Determination of definition information size of a plug-in function

The formula provided below is used to determine the definition information size of a plug-in function:

Formula

$0.6 + c + 0.2 \times d$ (kilobytes)

c:

Total number of plug-in functions used in DML by a single plug-in#:

d:

Total number of parameters of plug-in functions used in DML by a single plug-in#:

Note

This formula is for one plug-in. If multiple plug-ins are installed, use the total for all of the installed plug-ins.

#: Use the following SQL code to determine the total number of plug-in functions used in DML and the parameters of the plug-in functions used in DML:

```
SELECT COUNT(*),SUM(N_PARAM) FROM MASTER.SQL_PLUGIN_ROUTINES
  WHERE PLUGIN_NAME = 'plug-in-name'
  AND (TIMING_DESCRIPTOR = 'ADT_FUNCTION'
    OR TIMING_DESCRIPTOR IS NULL
    OR TIMING_DESCRIPTOR = 'BEFORE_INSERT'
    OR TIMING_DESCRIPTOR = 'AFTER_INSERT'
    OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE'
    OR TIMING_DESCRIPTOR = 'AFTER_UPDATE'
    OR TIMING_DESCRIPTOR = 'BEFORE_DELETE'
    OR TIMING_DESCRIPTOR = 'AFTER_DELETE'
    OR TIMING_DESCRIPTOR = 'BEFORE_PURGE_TABLE'
    OR TIMING_DESCRIPTOR = 'AFTER_PURGE_TABLE'
    OR TIMING_DESCRIPTOR = 'INDEX_SEARCH'
    OR TIMING_DESCRIPTOR = 'INDEX_COUNT'
    OR TIMING_DESCRIPTOR = 'INDEX_INSERT'
    OR TIMING_DESCRIPTOR = 'INDEX_BEFORE_UPDATE'
    OR TIMING_DESCRIPTOR = 'INDEX_AFTER_UPDATE'
    OR TIMING_DESCRIPTOR = 'INDEX_DELETE'
    OR TIMING_DESCRIPTOR = 'PURGE_INDEX'
    OR TIMING_DESCRIPTOR = 'INDEX_MAINTENANCE_DEFERRED'
    OR TIMING_DESCRIPTOR = 'BEFORE_INSERT_DC'
    OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE_DC'
    OR TIMING_DESCRIPTOR = 'BEFORE_DATA_CHECK'
    OR TIMING_DESCRIPTOR = 'AFTER_DATA_CHECK')
```

(3) Determination of each definition scalar function's definition information size for each function

The table below shows the definition information size for each system definition scalar function:

Table C–1: Definition information size for each system definition scalar function

Functions	Definition information size (kilobytes)
ACOS, ADD_INTERVAL, ASCII, ASIN, ATAN, ATAN2, CENTURY, COS, COSH, CHR, DATE_TIME, DAYNAME, DAYOFWEEK, DAYOFYEAR, DEGREES, EXP, INTERVAL_DATETIMES, LAST_DAY, LN, LOG10, MIDNIGHTSECONDS, MONTHNAME, MONTHS_BETWEEN, NEXT_DAY, PI, RADIANS, SIN, SINH, SQRT, TAN, TANH, WEEK, WEEKOFMONTH, YEARS_BETWEEN	2
POWER, IS_DBLBYTES, IS_SNGLBYTES, ISDIGITS, ROUNDMONTH, TRANSL_LONG	4
CEIL, FLOOR, HALF, INSERTSTR, INSERTSTR_LONG, LEFTSTR, LTRIMSTR, NUMEDIT, QUARTER, REPLACE_LONG, REVERSESTR, RIGHTSTR, RTRIMSTR, SIGN, STRTONUM, TRUNCYEAR	6
LTRIM, REPLACE, RTRIM, TRANSL, TRUNC	12
POSSTR, ROUND	18
GREATEST, LEAST	32

D. Determining the Number of Locked Resources

This appendix provides and explains formulas for determining the number of locked resources needed to execute various SQL statements, utilities, and commands. The maximum number of locked resources is determined, some buffer space is added, and a pool size is set in the following operands:

- `pd_lck_pool_size` operand in each server's server definition
- `pd_fes_lck_pool_size` operand in the front-end server definition

Notes

1. The number of locked resources count determined here is valid within a transaction. When multiple SQLs are executed in a single transaction, the total of their locked resources is needed. However, it is not needed if locking has already been applied. For details about locking, see the *HiRDB Version 9 UAP Development Guide*.
2. For a HiRDB/Parallel Server, the number of resources for each back-end server must be computed in terms of the number of resources (RDAREAs, indexes, lines) managed by the applicable back-end server.

D.1 Definition SQLs

(1) ALLOCATE MEMORY TABLE

For `ALLOCATE MEMORY TABLE`, determine the value if you will be using a memory-resident database.

(a) For HiRDB/Single Server

9 + *number of table columns* + *number of indexes defined for the table* × 2 + *number of tables subject to memory database processing* + *number of indexes having the same name as the indexes defined for the table* + *number of indexes stored in the same database area as for the specified index database area*

Add the following if there are routines that become invalid:

+ *number of routines in which the object becomes invalid* × 3 + 2

Add the following if there are triggers that become invalid:

+ *number of triggers for which the object becomes invalid* × 2 + 2 + *number of trigger event columns for which the object becomes invalid* + 1 + *number of parameters used in the triggers for which the object becomes invalid* + 1

(b) For HiRDB/Parallel Server (front-end server)

1

Add the following if there are routines that become invalid:

+ *number of routines in which the object becomes invalid*

(c) For HiRDB/Parallel Server (dictionary server)

8 + *number of table columns* + *number of indexes defined for the table* × 2 + *number of tables subject to memory database processing that is stored in the same XDS* + *number of indexes having the same name as the indexes defined for the table* + *number of indexes stored in the same database area as for the specified index database area*

Add the following if there are routines that become invalid:

+ *number of routines in which the object becomes invalid* × 2 + 2

Add the following if there are triggers that become invalid:

+ *number of triggers for which the object becomes invalid* × 2 + 2 + *number of trigger event columns for which the object becomes invalid* + 1 + *number of parameters used in the triggers for which the object becomes invalid* + 1

(2) ALTER INDEX

(a) For HiRDB/Single Server

5 + (*number of index member columns* × 2) + *number of procedures and triggers whose objects become invalid*

Add the following if there is a procedure whose objects become invalid:

+ *number of procedures whose objects become invalid* × 3 + *number of resources used by procedures whose objects become invalid* × 5

Add the following if there is a trigger whose objects become invalid:

+ *number of triggers whose objects become invalid* × 5 +*number of columns specified as the trigger event of an UPDATE trigger* × 3 + *number of trigger action procedures that reference tables for which an index is defined* × 5

Add the following if there is an old or new value correlation name:

+ 4

Add the following if statistical information regarding the index is output:

+ 2

Add the following if it is a partitioning key index:

+ *number of partitioning conditions* × 2

Add the following if it is an index that uses index types:

+ 1 + *number of abstract data types that use types that define the index type* × 2 + *number of functions that use the index type* × 2

Add the following if it is a substructure index:

+ *number of member substructure paths* × 2

Add the following if an exception value is specified:

+ 2

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of procedures and triggers whose objects become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

4 + (*number of index member columns* × 2)

Add the following if there is a procedure whose objects become invalid:

+ *number of procedures whose objects become invalid* × 3 + *number of resources used by procedures whose objects become invalid* × 5

Add the following if there is a trigger whose objects become invalid:

+ *number of triggers whose objects become invalid* × 5 + *number of columns specified as the trigger event of an UPDATE trigger* × 3 + *number of trigger action procedures that reference the table for which the index is defined* × 5

Add the following if there is an old or new value correlation name:

+ 4

Add the following if statistical information regarding the index is output:

+ 2

Add the following if it is a partitioning key index:

+ *number of partitioning conditions* × 2

Add the following if it is an index that uses the index type:

+ 1 + *number of abstract data types that use types that define the index type* × 2 + *number of functions that use the index type* × 2

Add the following if it is a substructure index:

+ *number of member substructure paths* × 2

Add the following if an exception value is specified:

+ 2

(3) ALTER PROCEDURE

(a) For HiRDB/Single Server

5 + *number of tables accessed in SQL statement preprocessing* + *number of view tables accessed in SQL statement preprocessing* + *number of base tables that serve as the base for view tables accessed in SQL statement preprocessing*

Add the following if the executer is not the owner:

+ 2

Add the following if a procedure name is specified:

+ 2

Add the following if AUTHORIZATION is specified:

+ *number of invalid routines owned by a target person* + *number of invalid PUBLIC routines defined by a target person* + 2

Also add the following if ALL is specified:

+ *total number of routines owned by a target person* + *total number of* `PUBLIC` *routines defined by a target person* + 1

Also add the following if INDEX USING is specified:

+ *number of routines that use the specified table as a resource* × 2 + 3

Add the following if AUTHORIZATION is not specified:

+ *total number of invalid routines in the system*

Also add the following if ALL is specified:

+ *total number of routines in the system*

Also add the following if INDEX USING is specified:

+ *number of routines that use the specified table as a resource* × 2 + 3

Add the following if there are resources used in routines:

+ *number of routines used* × 2 + 1

Add the following if there are tables in the resources used:

+ *number of tables used* + 1

Add the following if there are view tables in the resources used:

+ *number of view tables used* + 1 + *number of utilized resources used in view tables used* + 1

Add the following if there are indexes in the resources to be used:

+ *number of indexes to be used* + 1

Add the following if there are routines in the resources to be used:

+ *number of routines used* + 1

Add the following if there are user-defined types in the resources to be used:

+ *user-defined types to be used* + 1

Add the following if parameters are specified:

+ *number of parameters* + 1

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of tables accessed in SQL statement preprocessing* + *number of view tables accessed in SQL statement preprocessing* + *number of base tables that serve as the base for view tables accessed in SQL statement preprocessing*

(c) For HiRDB/Parallel Server (dictionary server)

5

Add the following if the executer is not the owner:

+ 2

Add the following if a procedure name is specified:

+ 2

Add the following if AUTHORIZATION is specified:

+ *number of invalid routines owned by a target person* + *number of invalid* `PUBLIC` *routines defined by a target person*

Also add the following if ALL is specified:

+ *total number of routines owned by a target person* + *total number of* `PUBLIC` *routines defined by a target person*

Also add the following if INDEX USING is specified:

+ *number of routines that use the specified table as a resource* × 2 + 3

Add the following if AUTHORIZATION is not specified:

+ *total number of invalid routines in system*

Also add the following if ALL is specified:

+ *total number of routines in system*

Also add the following if INDEX USING is specified:

+ *number of routines that use the specified table as a resource* × 2 + 3

Add the following if there are resources used in routines:

+ *number of routines used* × 2 + 1

Add the following if there are tables in the resources used:

+ *number of tables used* + 1

Add the following if there are view tables in the resources used:

+ *number of view tables used* + 1 + *number of utilized resources used in view tables used* + 1

Add the following if there are indexes in the resources to be used:

+ *number of indexes to be used* + 1

Add the following if there are routines in the resources to be used:

+ *number of routines used* + 1

Add the following if there are user-defined types in the resources to be used:

+ *user-defined types to be used* + 1

Add the following if parameters are specified:

+ *number of parameters* + 1

(d) For HiRDB/Parallel Server (back-end server)

1

(4) ALTER ROUTINE

(a) For HiRDB/Single Server

5 + *number of tables accessed in SQL statement preprocessing* + *number of view tables accessed in SQL statement preprocessing* + *number of base tables that serve as the base for view tables accessed in SQL statement preprocessing*

Add the following if there are triggers:

+ 1

Add the following if the executer is not the owner:

+ 2

Add the following if AUTHORIZATION is specified:

+ *number of invalid routines owned by a target person* + *number of invalid* `PUBLIC` *routines defined by a target person* + 1

Also add the following if ALL is specified:

+ *total number of routines owned by a target person* + *total number of* `PUBLIC` *routines defined by a target person*

Also add the following if there are triggers owned by a target person:

+ *number of triggers owned by a target person* × 4 + 3 + *number of tables that define triggers owned by a target person* + *number of trigger event columns* × 2 + 1 + *number of resource information items used in trigger action conditions* × 2 + 1 + *number of parameters used by triggers* × 2 + 1

Add the following if AUTHORIZATION is not specified:

+ *total number of invalid routines in the system*

Also add the following if ALL is specified:

+ *total number of routines in the system*

Also add the following if there are triggers in the system:

+ *total number of triggers* × 4 + 3 + *total number of tables that define triggers* + *number of trigger event columns* × 2 + 1 + *number of resource information items used in trigger action conditions* × 2 + 1 + *number of parameters used by triggers* × 2 + 1

Add the following if there are resources used in routines:

+ *number of routines used* × 2

Add the following if there are tables in the resources used:

+ *number of tables used* + 1

Add the following if there are view tables in the resources used:

+ *number of view tables used* + 1 + *number of utilized resources used in view tables used* + 1

Add the following if there are indexes in the resources to be used:

+ *number of indexes to be used* + 1

Add the following if there are routines in the resources to be used:

+ *number of routines used* + 1

Add the following if there are user-defined types in the resources to be used:

+ *user-defined types to be used* + 1

Add the following if parameters are specified:

+ *number of parameters* + 1

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of tables accessed in SQL statement preprocessing* + *number of view tables accessed in SQL statement preprocessing* + *number of base tables that serve as the base for view tables accessed in SQL statement preprocessing*

Add the following if there are triggers:

+ 1

(c) For HiRDB/Parallel Server (dictionary server)

5

Add the following if the executer is not the owner:

+ 2

Add the following if AUTHORIZATION is specified:

+ *number of invalid routines owned by a target person* + *number of invalid* `PUBLIC` *routines defined by a target person*

Also add the following if ALL is specified:

+ *total number of routines owned by a target person* + *total number of* `PUBLIC` *routines defined by a target person*

Also add the following if there are triggers owned by a target person:

+ *number of triggers owned by a target person* × 4 + 3 + *number of tables that define triggers owned by a target person* + 1 + *number of trigger event columns* × 2 + 1 + *number of resource information items used in trigger action conditions* × 2 + 1 + *number of parameters used by triggers* × 2 + 1

Add the following if AUTHORIZATION is not specified:

+ *total number of invalid routines in the system*

Also add the following if ALL is specified:

+ *total number of routines in the system*

Also add the following if there are triggers in the system:

+ *total number of triggers* × 4 + 3 + *total number of tables that define triggers* + 1 + *number of trigger event columns* × 2 + 1 + *number of resource information items used in trigger action conditions* × 2 + 1 + *number of parameters used by triggers* × 2 + 1

Add the following if there are resources used in routines:

+ *number of routines used* × 2

Add the following if there are tables in the resources used:

+ *number of tables used* + 1

Add the following if there are view tables in the resources used:

+ *number of view tables used* + 1 + *number of utilized resources used in view tables used* + 1

Add the following if there are indexes in the resources to be used:

+ *number of indexes to be used* + 1

Add the following if there are routines in the resources to be used:

+ *number of routines used* + 1

Add the following if there are user-defined types in the resources to be used:

+ *user-defined types to be used* + 1

Add the following if parameters are specified:

+ *number of parameters* + 1

(d) For HiRDB/Parallel Server (back-end server)

1

(5) ALTER TABLE

(a) For HiRDB/Single Server (operations other than the addition or deletion of primary keys)

4 + *number of view tables defined based on the table*

Add the following if there are routines that become invalid:

+ *number of routines whose objects become invalid* × 3 + 2

Add the following if triggers are defined:

+ *number of trigger definitions* × 2 + 2 + *number of defined trigger event columns* + 1 + *number of parameters used by defined triggers* + 1

Add the following if specifying ADD column-name:

+ 5

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if adding BLOB columns:

+ *number of RDAREAs added to the table* + 1 + *number of RDAREAs used by the table* + 1 + *number of table BLOB columns* + 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*

- **Also add the following if there are routines that use the table as a resource:**

  + *number of routines that use the table as a resource* + 1
- **Also add the following if there are view tables defined based on the table:**

  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if indexes are defined:**

  + *number of indexes defined in the table* + 1
- **Also add the following if the table is a partitioned table:**

  + *number of table partitions* × 2 + 2

Also add the following if partitioning key indexes are defined:

+ *number of partitioning key indexes* × *number of table partitions* + 1

Also add the following when BLOB columns are defined:

+ *number of table BLOB columns* × *number of table partitions*

Also add the following if it is a matrix-partitioned table:

+ 3

Also add the following if adding user-defined type columns:

+ 4 + *number of attributes used by user-defined types*

- **Also add the following if there are routines that use the table as a resource:**

  + *number of routines that use the table as a resource* + 1
- **Also add the following if there are view tables defined based on the table:**

  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if there are user-defined types for which the BLOB attribute is defined:**

  + *number of RDAREAs added to the table* + 1 + *number of RDAREAs used by the able* + 1 + *number of table BLOB columns* + 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*

  Also add the following if indexes are defined:

  + *number of indexes defined in the table* + 1

  Also add the following if the table is a partitioned table:

  + *number of table partitions* × 2 + 2

  ● Also add the following if partitioning key indexes are defined:

  + *number of partitioning key indexes* × *number of table partitions* + 1

  ● Also add the following when BLOB columns are defined:

  + *number of table BLOB columns* × *number of table partitions*

  ● Also add the following if it is a matrix-partitioned table:

  + 3

  Also add the following if abstract data types of plug-ins are used:

  + *number of plug-ins used* + 1 + *number of routines of plug-ins used* + 1 + *number of parameters of plug-in routines used* + 1

Also add the following if the table is a FIX table:

+ *number of RDAREAs used by the table*

- **Also add the following if there are routines that use the table as a resource:**

  + *number of routines that use the table as a resource* + 1
- **Also add the following if there are view tables defined based on the table:**

  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if the table is a partitioned table:**

  + *number of table partitions* × 2 + 2

Also add the following if adding a NOT NULL column to a non-FIX table:

+ *number of RDAREAs used by the table*

- **Also add the following if there are routines that use the table as resources:**

  + *number of routines that use the table as resources* + 1
- **Also add the following if there are view tables defined based on the table:**

  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if the table is a partitioned table:**

  + *number of table partitions* × 2 + 2

Also add the following if adding a column that has a DEFAULT clause specification:

+ *number of RDAREAs used by the table*

- **Also add the following if there are routines that use the table as a resource:**

  + *number of routines that use the table as a resource* + 1

- **Also add the following if there are view tables defined based on the table:**
  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if the table is a partitioned table:**
  + *number of table partitions* × 2 + 2

Also add the following if columns are extracted:

+ 1

Add the following if there is an ADD RDAREA:

+ 9 + *number of member columns in the table* + 2 + *number of indexes defined in the table* + 4 + *number of indexes defined in the table* × 2 + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*

Add the following if it is a FIX hash table:

+ *number of table-dedicated RDAREAs used by the table* + *number of table partitions*

Also add the following if adding RDAREAs that are already being used for tables:

+ *number of RDAREAs already being used*

Also add the following if adding RDAREAs not being used for tables:

+ *number of RDAREAs added* + 1

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1 + *number of RDAREAs used by indexes defined in the table* × 2 + 1

- **Also add the following if adding RDAREAs not being used for tables:**
  + *number of indexes defined in the table*
- **Also add the following if partitioning key indexes are defined:**
  + *number of partitioning key indexes* × *number of partitions after RDAREAs are added* + 1
- **Also add the following if plug-in indexes are defined:**
  + 2 + *number of plug-in columns* + 1 + *number of attributes of plug-ins used by the table* + 1 + *number of routines of plug-ins used by the table* + 1 + *number of parameters of plug-in routines used by the table* + 1 + *number of BLOB attributes of plug-ins used by the table* × *number of partitions after RDAREAs are added*
- **Also add the following if substructure indexes are defined:**
  + *number of substructure indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* + 1 + *number of BLOB columns defined in the table* × *number of partitions after RDAREAs are added* + 1 + *number of RDAREAs used by BLOB columns defined in the table* × 2 + 1

- **Also add the following if adding RDAREAs not being used for tables:**
  + *number of BLOB columns defined in the table*

Also add the following if user-defined type columns are defined in the table:

+ *number of user-defined type columns defined in the table* + 1 + *number of attributes of user-defined types used in the table* + 1

- **Also add the following if BLOB attributes are defined in user-defined types:**
  + *number of BLOB attributes of user-defined types used in the table* + 1 + *number of RDAREAs used by BLOB attributes of user-defined types used in the table* × 2 + 1 + *number of BLOB attributes of user-defined types used in the table* × *number of partitions after RDAREAs are added* + 1
- **Also add the following if adding RDAREAs not being used for tables:**
  + *number of user-defined type columns defined in the table*

Add the following if there is a CHANGE column-name:

+ 11

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3

Also add the following if triggers are defined in the table:

+ *number of triggers that use columns as resources* × 2

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource* + *number of routines that use columns as parameters* × 2

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table*

- **Also add the following if columns are index member columns:**

  Also add the following if the column data types are `VARCHAR`, `NVARCHAR`, or `MVARCHAR`:

  + *number of member columns in the table* + *number of index RDAREAs used by the table* + 1 + *number of member columns of indexes defined in the table* + 1

  ● Also add the following if partitioning key indexes are defined:

  + *number of partitioning key indexes* × *number of partitions* + 1

Also add the following if columns are BLOB columns:

+ 1

Also add the following if columns are user-defined type columns:

+ 2 + *number of attributes of user-defined types used by columns* + 1

Also add the following if NO SPLIT or RECOVERY is specified:

+ *number of RDAREAs used by the table*

- **Also add the following if it is a partitioned table:**

  + *number of table partitions* + 1

  Also add the following when BLOB columns are defined:

  + *number of BLOB columns defined in the table* × *number of partitions*

Also add the following if data size changes:

- **Also add the following if an index is defined in the table:**

  + 1

Also add the following if a column recovery restriction is defined:

- **Also add the following if columns are BLOB columns:**

  + 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*

Also add the following if a column is a user-defined type column that has the BLOB attribute:

+ 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*

Add the following if there is a CHANGE CLUSTER KEY:

+ 10 + *number of member columns of the cluster key index* × 2 + 1 + *number of RDAREAs used by the table*

Also add the following if there are routines with objects that become invalid:

+ *number of routines whose objects become invalid* × 2

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table*

- **Also add the following if partitioning key indexes are defined:**

  + *number of partitioning key indexes* × *number of partitions* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table*

- **Also add the following if it is a partitioned table:**

  + *number of BLOB columns defined in the table* × *number of partitions* + 1

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

- **Also add the following if it is a mixed hash and matrix partitioned table:**

  + 2

Also add the following if it is a FIX table:

+ *number of member columns in the table*

Add the following if there is a CHANGE LOCK:

+ 5 + *number of RDAREAs used by the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

Add the following if there is a CHANGE HASH:

+ 5 + *number of member columns in the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1

- **Also add the following if partitioning key indexes are defined:**
  + *number of partitioning key indexes* × *number of partitions* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* × *number of partitions* + 1

Also add the following if it is a FIX hash partitioned table:

+ *number of table partitions* + 1 + *number of RDAREAs used by the table* + 1

Add the following if there is a CHANGE SEGMENT REUSE:

+ 4 + 2 + *number of RDAREAs used by the table*

Also add the following if it is a partitioned table:

+ *number of table partitions* + *number of table partitioning conditions* × 2

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if the SEGMENT REUSE specification is changed:

+ *number of RDAREAs used by the table* + 1

- **Also add the following if indexes are defined in the table:**
  + *number of indexes defined in the table* + 1
  Also add the following if partitioning key indexes are defined:
  + *number of partitioning key indexes* × *number of partitions* + 1
- **Also add the following if BLOB columns are defined in the table:**
  + *number of BLOB columns defined in the table* × *number of partitions* + 1

Add the following if there is a CHANGE INSERT ONLY:

+ 5 + *number of RDAREAs used by the table*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3 + 1

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource* + 1

Also add the following if a row deletion period is specified:

+ 1

Add the following if there is a DROP column name:

+ 18 + *number of member columns in the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 4 + *number of member columns in view tables defined based on the table* + *number of resources used by view tables defined based on the table* + *number of users having access permission to view tables defined based on the table*

- **Also add the following if there is a public view table defined based on the table:**

  + 4

Also add the following if triggers are defined in the table:

+ *number of triggers that use columns as event columns* × 2 + *number of triggers defined in the table* × 4 +*number of resources used by triggers defined in the table* × 2 + *number of routines used by triggers defined in the table* + *number of parameters used by triggers defined in the table*

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource* × 2 + *number of routines that use columns as parameters* × 2

Also add the following if statistical information relating to columns is output:

+ 2

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table*

- **Also add the following if columns are index member columns:**

  + *number of indexes that use columns as member columns* × 2 + 2 + *number of index RDAREAs used by the table* + 1 + *number of member columns of indexes that use columns as member columns* + 1

  Also add the following if partitioning key indexes are defined:

  + *number of partitioning key indexes* × *number of table partitions* + 1

  Also add the following if statistical information relating to indexes is output:

  + *number of indexes that use columns as member columns* + 1

  Also add the following if exception values are specified for indexes:

  + *number of exception values of indexes that use columns as member columns* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table*

- **Also add the following if it is a partitioned table:**

  + *number of BLOB columns defined in the table* × *number of table partitions*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

- **Also add the following if it is a matrix-partitioned table:**

  + 2

Also add the following if the facility for predicting reorganization time is used:

+ *number of index RDAREAs no longer used by the table* × 2 + 2 + *number of RDAREAs in which the table and indexes are no longer stored* × 2

Add the following if there is a RENAME TABLE:

+ 7 + *number of member columns in the table* + *number of users with permission to access the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 2

Also add the following if statistical information relating to the table is output:

+ 2

Also add the following if statistical information relating to columns is output:

+ *number of columns for which statistical information exists* + 1

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

- **Also add the following if it is a row partitioned table:**

  + *number of table partitioning conditions* × 2 + 1

- **Also add the following if it is a matrix-partitioned table:**
  + *number of table partitioning conditions* + 4
  Also add the following if it is a mixed hash and matrix partitioned table:
  + *number of table partitioning keys* + 1
- **Also add the following if indexes are defined in the table:**
  + *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1
  Also add the following if partitioning key indexes are defined:
  + *number of partitioning key indexes* × *number of partitions* + 1
  Also add the following if substructure indexes are defined:
  + *number of substructure indexes* + 1
  Also add the following if plug-in indexes are defined:
  + *number of functions that plug-in indexes use* + 1
  Also add the following if exception values are specified for indexes:
  + *number of exception values of indexes that are defined in the table* + 1
  Also add the following if statistical information relating to indexes is output:
  + *number of indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:
+ *number of BLOB columns defined in the table* + 1

- **Also add the following if it is a partitioned table:**
  + *number of BLOB columns defined in the table* × *number of partitions* + 1

Also add the following if user-defined type columns are defined in the table:
+ *number of user-defined type columns defined in the table* + 1

- **Also add the following if indexes that use index types in the table are defined:**
  + *number of user-defined types to be used in indexes* + 1
- **Also add the following if BLOB attributes are defined in user-defined types:**
  + *number of BLOB attributes used in user-defined types* + 1
  Also add the following if it is a partitioned table:
  + *number of BLOB attributes used in user-defined types* × *number of partitions*

Also add the following if it is a public view table with a name that is the same as the table name before modification:
+ 3
Also add the following if it is a public view table with a name that is the same as the table name after modification:
+ 3

Add the following if there is a RENAME COLUMN:
+ 7 + *number of member columns in the table*
Also add the following if there are view tables defined based on the table:
+ *number of view tables defined based on the table* × 2
Also add the following if statistical information relating to columns is output:
+ 2
Also add the following if there are routines that use the table as a resource:
+ *number of routines that use the table as a resource* × 2 + 1 + *number of routines that use columns as parameters* × 2
Also add the following if it is a matrix-partitioned table:
+ 2
Also add the following if columns are index member columns:
+ *number of indexes that use columns as member columns* + 1

- **Also add the following if there are columns that are plug-in index member columns:**
  + *number of functions that plug-in indexes use* + 1

Also add the following if columns are BLOB columns:

+ 2

- **Also add the following if it is a partitioned table:**
  + *number of partitions*

Also add the following if columns are user-defined type columns:

- **Also add the following if there are columns that are index member columns that use the index type:**
  + *number of user-defined types to be used in indexes* + 1
- **Also add the following if BLOB attributes are defined in user-defined types:**
  + *number of BLOB attributes used in user-defined types* + 1
  Also add the following if it is a partitioned table:
  + *number of BLOB attributes used in user-defined types* × *number of partitions*

Add the following if there is a CHANGE RDAREA for a row partitioning table:

+ 8 + *number of member columns in the table* + *number of partitioning conditions before modification* × 2 + *number of partitioning conditions after modification* × 2 + *number of partitions before modification* + *number of partitions after modification* + *number of RDAREAs used by the table before modification* + *number of RDAREAs used by the table after modification* × 2 + 4 + *number of indexes defined in the table* + *number of indexes subject to deletion* + (2 + *number of indexes defined in the table* × 2 + *number of BLOB columns defined in the table*) × (*number of RDAREAs added to the table* + *number of RDAREAs to be deleted from the table*)

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1 + *number of indexes defined in the table* × *number of partitions before modification* + 1 + *number of indexes defined in the table* × *number of partitions after modification*

- **Also add the following if substructure indexes are defined:**
  + *number of substructure indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* × *number of partitions before modification* + 1 + *number of BLOB columns defined in the table* × *number of partitions after modification*

Also add the following if a referential constraint is defined that makes the table a referenced table:

+ *number of referential constraints that make the table a referenced table* + 1 + *number of referencing tables that make the table a referenced table*

Also add the following if a referential constraint is defined that makes the table a referencing table:

+ *number of referential constraints of the table* + 1

Also add the following if a check constraint is defined in the table:

+ *number of check constraints in the table* + 1

Also add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs no longer used by the table* × 2 + 2 + *number of RDAREAs whose data will be deleted* + *number of RDAREAs in which the table and indexes are no longer stored* × 2

Also add the following if there is a CHANGE RDAREA for a matrix-partitioned table:

+ 11

+ *number of member columns in the table* + *number of partitioning conditions before modification* + *number of partitioning conditions after modification* + *number of partitions before modification* + *number of partitions after modification* + *number of RDAREAs used by the table before modification*

+ *number of RDAREAs used by the table after modification* × 2 + 4

+ *number of indexes defined in the table* + *number of indexes subject to deletion* + (2 + *number of indexes defined in the table* × 2 + *number of BLOB columns defined in the table*) × (*number of RDAREAs added to the table* + *number of RDAREAs to be deleted from the table*)

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the tables as a resource*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1 + *number of indexes defined in the table* × *number of partitions before modification* + 1 + *number of indexes defined in the table* × *number of partitions after modification*

- **Also add the following if substructure indexes are defined:**

  + *number of substructure indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* × *number of partitions before modification* + 1 + *number of BLOB columns defined in the table* × *number of partitions after modification*

Also add the following if a referential constraint is defined that makes the table a referenced table:

+ *number of referential constraints that make the table a referenced table* + 1 + *number of referencing tables that make the table a referenced table*

Also add the following if a referential constraint is defined that makes the table a referencing table:

+ *number of referential constraints of the table* + 1

Also add the following if a check constraint is defined in the table:

+ *number of check constraints in the table* + 1

Also add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs no longer used by the table* × 2 + 2 + *number of RDAREAs whose data will be deleted* + *number of RDAREAs in which the table and indexes are no longer stored* × 2

(b) For HiRDB/Single Server (addition and deletion of primary keys)

(i) Addition of primary keys

15 + *number of columns in the target table* + *number of member columns of the primary key* × 2

+ *number of RDAREAs for index* × 3 + *number of RDAREAs for table* × 2

+ 5 + 2 × *number of RDAREAs for index*

Also add the following if the target table is a matrix-partitioned table:

+ 3 + *number of table partitioning information items*

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table* × 3 + 1

Also add the following if there are trigger action procedures that reference the target table:

+ *number of trigger action procedures that reference the target table* + 1

(ii) Deletion of primary keys

15 + *number of member columns of the primary key* × 2 + *number of RDAREAs for index* × 4

+ *number of RDAREAs for index* × 7 + *number of index segments*

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table* × 3 + 1

Also add the following if there are trigger action procedures that reference the target table:

+ *number of trigger action procedures that reference the target table* + 1

Also add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 2 + 1

+ *number of resources held by a routine that becomes invalid* × *number of routines that become invalid*

Also add the following if there are triggers that become invalid:

+ *number of triggers that become invalid* + 1

+ *number of resources referenced in the trigger action conditions that become invalid* + 1

+ *number of parameters in the trigger action procedures of the triggers that become invalid* + 1

+ *number of trigger event column information items for the triggers that become invalid* + 1

Also add the following if index optimization information is collected:

+ 2

Add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs for index* × 62 + 1

(c) For HiRDB/Parallel Server (front-end server) (operations other than the addition or deletion of primary keys)

1 + *number of routines whose objects become invalid* + *number of view tables defined based on the table*

Add the following if there is a DROP COLUMN:

+ *number of triggers defined in the table*

(d) For HiRDB/Parallel Server (front-end server) (addition and deletion of primary keys)

(i) Addition of primary keys

1

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table*

(ii) Deletion of primary keys

1

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table*

Also add the following if there are routines that become invalid:

+ *number of routines that become invalid*

(e) For HiRDB/Parallel Server (dictionary server) (operations other than the addition or deletion of primary keys)

1

Add the following if there are routines that become invalid:

+ *number of routines whose objects become invalid* × 3 + 2

Add the following if triggers are defined:

+ *number of trigger definitions* × 2 + 2 + *number of defined trigger event columns* + 1 + *number of parameters used by defined triggers* + 1

Also add the following if there is a DROP COLUMN:

+ *number of triggers defined in the table*

Add the following if there is an ADD column-name:

+ 5

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if adding BLOB columns:

+ *number of RDAREAs added to the table* + 1 + *number of RDAREAs used by the table* + 1 + *number of table BLOB columns*

- **Also add the following if there are routines that use the table as a resource:**

  + *number of routines that use the table as a resource* + 1

- **Also add the following if there are view tables defined based on the table:**

  + *number of view tables defined based on the table* × 2 + 2

- **Also add the following if indexes are defined:**

  + *number of indexes defined in the table* + 1

  Also add the following if the table is a partitioned table:

  + *number of table partitions* × 2 + 2

  Also add the following if partitioning key indexes are defined:

  + *number of partitioning key indexes* × *number of table partitions* + 1

  Also add the following when BLOB columns are defined:

  + *number of table BLOB columns* × *number of table partitions*

  Also add the following if it is a matrix-partitioned table:

  + 3

Also add the following if adding user-defined type columns:

+ 4 + *number of attributes used by user-defined types*

- **Also add the following if there are routines that use the table as a resource:**
  + *number of routines that use the able as a resource* + 1
- **Also add the following if there are view tables defined based on the table:**
  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if there are user-defined types for which BLOB attributes are defined:**
  + *number of RDAREAs added to the table* + 1 + *number of RDAREAs used by the table* + 1 + *number of table BLOB columns*
  Also add the following if indexes are defined:
  + *number of indexes defined in the table* + 1
  Also add the following if the table is a partitioned table:
  + *number of table partitions* × 2 + 2
  ● Also add the following if partitioning key indexes are defined:
  + *number of partitioning key indexes* × *number of table partitions* + 1
  ● Also add the following when BLOB columns are defined:
  + *number of table BLOB columns* × *number of table partitions*
  ● Also add the following if it is a matrix-partitioned table:
  + 3
  Also add the following if abstract data types of plug-ins are used:
  + *number of plug-ins used* + 1 + *number of routines of plug-ins used* + 1 + *number of parameters of plug-in routines used* + 1

Also add the following if the table is a FIX table:
+ *number of RDAREAs used by the table*

- **Also add the following if there are routines that use the table as a resource:**
  + *number of routines that use the table as a resource* + 1
- **Also add the following if there are view tables defined based on the table:**
  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if the table is a partitioned table:**
  + *number of table partitions* × 2 + 2

Also add the following if adding a NOT NULL column to a non-FIX table:
+ *number of RDAREAs used by the table*

- **Also add the following if there are routines that use the table as a resource:**
  + *number of routines that use the table as a resource* + 1
- **Also add the following if there are view tables defined based on the table:**
  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if the table is a partitioned table:**
  + *number of table partitions* × 2 + 2
- **Also add the following if columns are extracted:**
  + 1

Also add the following if adding a column that has a DEFAULT clause specification:
+ *number of RDAREAs used by the table*

- **Also add the following if there are routines that use the table as a resource:**
  + *number of routines that use the table as a resource* + 1
- **Also add the following if there are view tables defined based on the table:**
  + *number of view tables defined based on the table* × 2 + 2
- **Also add the following if the table is a partitioned table:**
  + *number of table partitions* × 2 + 2

Add the following if there is an ADD RDAREA:
+ 9 + *number of member columns in the table* + 2 + *number of indexes defined in the table*
Also add the following if it is a FIX hash table:

+ *number of table-dedicated RDAREAs used by the table* + *number of table partitions*

Also add the following if adding RDAREAs that are already being used for tables:

+ *number of RDAREAs already being used*

Also add the following if adding RDAREAs not being used for tables:

+ *number of RDAREAs added* + 1

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1 + *number of RDAREAs used by indexes defined in the table* × 2 + 1

- **Also add the following if adding RDAREAs not being used for tables:**

  + *number of indexes defined in the table*

- **Also add the following if partitioning key indexes are defined:**

  + *number of partitioning key indexes* × *number of partitions after RDAREAs are added* + 1

- **Also add the following if plug-in indexes are defined:**

  + 2 + *number of plug-in columns* + 1 + *number of attributes of plug-ins used by the table* + 1 + *number of routines of plug-ins used by the table* + 1 + *number of parameters of plug-in routines used by the table* + 1 + *number of BLOB attributes of plug-ins used by the table* × *number of partitions after RDAREAs are added*

- **Also add the following if substructure indexes are defined:**

  + *number of substructure indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* + 1 + *number of BLOB columns defined in the table* × *number of partitions after RDAREAs are added* + 1 + *number of RDAREAs used by BLOB columns defined in the table* × 2 + 1

- **Also add the following if adding RDAREAs not being used for tables:**

  + *number of BLOB columns defined in the table*

Also add the following if user-defined type columns are defined in the table:

+ *number of user-defined type columns defined in the table* + 1 + *number of attributes of user-defined types used in the table* + 1

- **Also add the following if BLOB attributes are defined in user-defined types:**

  + *number of BLOB attributes of user-defined types used in the table* + 1 + *number of RDAREAs used by BLOB attributes of user-defined types used in the table* × 2 + 1 + *number of BLOB attributes of user-defined types used in the table* × *number of partitions after RDAREAs are added* + 1

- **Also add the following if adding RDAREAs not being used for tables:**

  + *number of user-defined type columns defined in the table*

Add the following if there is a CHANGE column-name:

+ 11

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3

Also add the following if triggers are defined in the table:

+ *number of triggers that use columns as resources* × 2

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource* + *number of routines that use columns as parameters* × 2

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table*

- **Also add the following if columns are index member columns:**

  Also add the following if the column data types are `VARCHAR`, `NVARCHAR`, or `MVARCHAR`:

  + *number of member columns in the table* + *number of index RDAREAs used by the table* + 1 + *number of member columns of indexes defined in the table* + 1

  ● Also add the following if partitioning key indexes are defined:

  + *number of partitioning key indexes* × *number of partitions* + 1

Also add the following if columns are BLOB columns:

+ 1

Also add the following if columns are user-defined type columns:

+ 2 + *number of attributes of user-defined types used by column* + 1

Also add the following if NO SPLIT or RECOVERY is specified:

+ *number of RDAREAs used by the table*

- **Also add the following if it is a partitioned table:**

  + *number of table partitions* + 1

  Also add the following when BLOB columns are defined:

  + *number of BLOB columns defined in the table* × *number of partitions*

Also add the following if data size changes:

- **Also add the following if an index is defined in the table:**

  + 1

Add the following if there is a CHANGE CLUSTER KEY:

+ 10 + *number of cluster key member columns* × 2 + 1 + *number of RDAREAs used by the table*

Also add the following if there are routines with objects that become invalid:

+ *number of routines whose objects become invalid* × 2

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table*

- **Also add the following if partitioning key indexes are defined:**

  + *number of partitioning key indexes* × *number of partitions* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table*

- **Also add the following if it is a partitioned table:**

  + *number of BLOB columns defined in the table* × *number of partitions* + 1

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

- **Also add the following if it is a mixed hash and matrix partitioned table:**

  + 2

Also add the following if it is a FIX table:

+ *number of member columns in the table*

Add the following if there is a CHANGE LOCK:

+ 5 + *number of RDAREAs used by the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

Add the following if there is a CHANGE HASH:

+ 5 + *number of member columns in the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1

- **Also add the following if partitioning key indexes are defined:**

  + *number of partitioning key indexes* × *number of partitions* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* × *number of partitions* + 1

Also add the following if it is a FIX hash partitioned table:

+ *number of table partitions* + 1 + *number of RDAREAs used by the table*+ 1

Add the following if there is a CHANGE SEGMENT REUSE:

+ 4

Also add the following if it is a partitioned table:

+ *number of table partitions* + *number of table partitioning conditions* × 2

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if SEGMENT REUSE specification is changed:

+ *number of RDAREAs used by the table* + 1

- **Also add the following if indexes are defined in the table:**

  + *number of indexes defined in the table* + 1

  Also add the following if partitioning key indexes are defined:

  + *number of partitioning key indexes* × *number of partitions* + 1

- **Also add the following if BLOB columns are defined in the table:**

  + *number of BLOB columns defined in the table* × *number of partitions* + 1

Add the following if there is a CHANGE INSERT ONLY:

+ 5 + *number of RDAREAs used by the table*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 3 + 1

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource* + 1

Also add the following if a row deletion period is specified:

+ 1

Add the following if there is a DROP column-name:

+ 18 + *number of member columns in the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 4 + *number of member columns in view tables defined based on the table* + *number of resources used by view tables defined based on the table* + *number of users having access permission to view tables defined based on the table*

- **Also add the following if there is a public view table defined based on the table:**

  + 4

Also add the following if triggers are defined in the table:

+ *number of triggers that use columns as event columns* × 2 + *number of triggers defined in the table* × 4 + *number of resources used by triggers defined in the table* × 2 + *number of routines used by triggers defined in the table* + *number of parameters used by triggers defined in the table*

Also add the following if there are routines that use the tables as a resource:

+ *number of routines that use the table as a resource* × 2 + *number of routines that use columns as parameters* × 2

Also add the following if statistical information relating to columns is output:

+ 2

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table*

- **Also add the following if columns are index member columns:**

+ *number of indexes that use columns as member columns* × 2 + 2 + *number of index RDAREAs used by the table* + 1 + *number of member columns of indexes that use columns as member columns* + 1

Also add the following if partitioning key indexes are defined:

+ *number of partitioning key indexes* × *number of table partitions* + 1

Also add the following if statistical information relating to indexes is output:

+ *number of indexes that use columns as member columns* + 1

Also add the following if exception values are specified for indexes:

+ *number of exception values of indexes that use columns as member columns* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table*

- **Also add the following if it is a partitioned table:**

  + *number of BLOB columns defined in the table* × *number of table partitions*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

- **Also add the following if it is a matrix-partitioned table:**

  + 2

Also add the following if the facility for predicting reorganization time is used:

+ *number of index RDAREAs no longer used by the table* × 2 + 2 + *number of RDAREAs in which the table and indexes are no longer stored* × 2

Add the following if there is a RENAME TABLE:

+ 7 + *number of member columns in the table* + *number of users with permission to access the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 2

Also add the following if statistical information relating to the table is output:

+ 2

Also add the following if statistical information relating to columns is output:

+ *number of columns for which statistical information exists* + 1

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if it is a partitioned table:

+ *number of table partitions* + 1

- **Also add the following if it is a row partitioned table:**

  + *number of table partitioning conditions* × 2 + 1

- **Also add the following if it is a matrix-partitioned table:**

  + *number of table partitioning conditions* + 4

  Also add the following if it is a mixed hash and matrix partitioned table:

  + *number of table partitioning keys* + 1

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1

- **Also add the following if partitioning key indexes are defined:**

  + *number of partitioning key indexes* × *number of partitions* + 1

- **Also add the following if substructure indexes are defined:**

  + *number of substructure indexes* + 1

- **Also add the following if plug-in indexes are defined:**

  + *number of functions that plug-in indexes use* + 1

- **Also add the following if exception values are specified for indexes:**

  + *number of exception values of indexes that are defined in the table* + 1

- **Also add the following if statistical information relating to indexes is output:**

  + *number of indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* + 1

- **Also add the following if it is a partitioned table:**

  + *number of BLOB columns defined in the table* × *number of partitions* + 1

Also add the following if user-defined type columns are defined in the table:

+ *number of user-defined type columns defined in the table* + 1

- **Also add the following if indexes that use index types in the table are defined:**

  + *number of user-defined types to be used in indexes* + 1

- **Also add the following if BLOB attributes are defined in user-defined types:**

  + *number of BLOB attributes used in user-defined types* + 1

  Also add the following if it is a partitioned table:

  + *number of BLOB attributes used in user-defined types* × *number of partitions*

Also add the following if it is a public view table with a name that is the same as the table name before modification:

+ 3

Also add the following if it is a public view table with a name that is the same as the table name after modification:

+ 3

Add the following if there is a RENAME COLUMN:

+ 7 + *number of member columns in the table*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table* × 2

Also add the following if statistical information relating to columns is output:

+ 2

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource* × 2 + 1 + *number of routines that use columns as parameters* × 2

Also add the following if it is a matrix-partitioned table:

+ 2

Also add the following if columns are index member columns:

+ *number of indexes that use columns as member columns* + 1

- **Also add the following if there are columns that are plug-in index member columns:**

  + *number of functions that plug-in indexes use* + 1

Also add the following if columns are BLOB columns:

+ 2

- **Also add the following if it is a partitioned table:**

  + *number of partitions*

Also add the following if columns are user-defined type columns:

- **Also add the following if there are columns that are index member columns that use the index type:**

  + *number of user-defined types to be used in indexes* + 1

- **Also add the following if BLOB attributes are defined in user-defined types:**

  + *number of BLOB attributes used in user-defined types* + 1

  Also add the following if it is a partitioned table:

  + *number of BLOB attributes used in user-defined types* × *number of partitions*

Add the following if there is a CHANGE RDAREA for a row-partitioned table:

+ 8 + *number of member columns in the table* + *number of partitioning conditions before modification* × 2 + *number of partitioning conditions after modification* × 2 + *number of partitions before modification* + *number of partitions after modification* + *number of RDAREAs used by the table before modification* + *number of RDAREAs used by the table after modification* × 2 + 2 + *number of indexes defined in the table*

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1 + *number of indexes defined in the table* × *number of partitions before modification* + 1 + *number of indexes defined in the table* × *number of partitions after modification*

- **Also add the following if substructure indexes are defined:**

  + *number of substructure indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* × *number of partitions before modification* + 1 + *number of BLOB columns defined in the table* × *number of partitions after modification*

Also add the following if a referential constraint is defined that makes the table a referenced table:

+ *number of referential constraints that make the table a referenced table* + 1 + *number of referencing tables that make the table a referenced table*

Also add the following if a referential constraint is defined that makes the table a referencing table:

+ *number of referential constraints of the table* + 1

Also add the following if a check constraint is defined in the table:

+ *number of check constraints of the table* + 1

Also add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs no longer used by the table* × 2 + 2 + *number of RDAREAs whose data will be deleted* + *number of RDAREAs in which the table and indexes are no longer stored* × 2

Add the following if there is a CHANGE RDAREA for a matrix-partitioned table:

+ 11 + *number of member columns in the table* + *number of partitioning conditions before modification* + *number of partitioning conditions after modification* + *number of partitions before modification* + *number of partitions after modification* + *number of RDAREAs used by the table before modification* + *number of RDAREAs used by the table after modification* × 2 + 2 + *number of indexes defined in the table*

Also add the following if there are routines that use the table as a resource:

+ *number of routines that use the table as a resource*

Also add the following if there are view tables defined based on the table:

+ *number of view tables defined based on the table*

Also add the following if indexes are defined in the table:

+ *number of indexes defined in the table* + 1 + *number of member columns of indexes defined in the table* + 1 + *number of indexes defined in the table* × *number of partitions before modification* + 1 + *number of indexes defined in the table* × *number of partitions after modification*

- **Also add the following if substructure indexes are defined:**

  + *number of substructure indexes defined in the table* + 1

Also add the following if BLOB columns are defined in the table:

+ *number of BLOB columns defined in the table* × *number of partitions before modification* + 1 + *number of BLOB columns defined in the table* × *number of partitions after modification*

Also add the following if a referential constraint is defined that makes the table a referenced table:

+ *number of referential constraints that make the table a referenced table* + 1 + *number of referencing tables that make the table a referenced table*

Also add the following if a referential constraint is defined that makes the table a referencing table:

+ *number of referential constraints of the table* + 1

Also add the following if a check constraint is defined in the table:

+ *number of check constraints of the table* + 1

Also add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs no longer used by the table* × 2 + 2 + *number of RDAREAs whose data will be deleted* + *number of RDAREAs in which the table and indexes are no longer stored* × 2

(f) For HiRDB/Parallel Server (dictionary server) (addition and deletion of primary keys)

(i) Addition of primary keys

9 + *number of columns in the target table* + *number of member columns of the primary key*
+ *number of RDAREAs for index* × 3 + *number of RDAREAs for table* × 2

Also add the following if the target table is a matrix-partitioned table:

+ 3 + *number of table partitioning information items*

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table* × 3 + 1

Also add the following if there are trigger action procedures that reference the target table:

+ *number of trigger action procedures that reference the target table* + 1

(ii) Deletion of primary keys

9 + *number of member columns of the primary key* × 2 + *number of RDAREAs for index* × 4

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table* × 2 + 1

Also add the following if there are trigger action procedures that reference the target table:

+ *number of trigger action procedures that reference the target table* + 1

Also add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 2 + 1
+ *number of resources held by a routine that becomes invalid* × *number of routines that become invalid*

Also add the following if there are triggers that become invalid:

+ *number of triggers that become invalid* + 1
+ *number of resources referenced in the trigger action conditions that become invalid* + 1
+ *number of parameters in the trigger action procedures of the triggers that become invalid* + 1
+ *number of trigger event column information items for the triggers that become invalid* + 1

Also add the following if index optimization information is collected:

+ 2

Add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs for index* × 62 + 1

(g) For HiRDB/Parallel Server (back-end server) (operations other than the addition or deletion of primary keys)

number of routines whose objects become invalid

Add the following if there is a DROP COLUMN:

+ *number of triggers defined in the table*

Add the following if there is an ADD column name:

Also add the following if adding BLOB columns:

+ 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*

Also add the following if adding user-defined type columns that have BLOB attributes:

+ 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*

Add the following if there is an ADD RDAREA:

+ 4 + *number of indexes defined in the table* × 2 + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*

Add the following if there is a CHANGE column-name:

Add the following if column recovery restrictions are defined:

- **Also add the following if columns are BLOB columns:**
  + 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*
- **Also add the following if a column is a user-defined type column that has a BLOB attribute:**

+ 2 + (1 + *number of indexes defined in the table* + *number of BLOB columns defined in the table* + *number of BLOB attributes of user-defined types defined in the table*) × *number of LOB RDAREAs used by the table*

Add the following if there is a CHANGE SEGMENT REUSE:

+ 2 + *number of RDAREAs used by the table*

Add the following if there is a CHANGE RDAREA:

+ 2 + *number of indexes subject to deletion* + (2 + *number of indexes defined in the table* × 2 + *number of BLOB columns defined in the table*) × (*number of RDAREAs added to the table*+ *number of RDAREAs to be deleted from the table*)

(h) HiRDB/Parallel Server (back-end server) (addition and deletion of primary keys)

(i) Addition of primary keys

5 + *number of RDAREAs for index* × 2

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table*

(ii) Deletion of primary keys

5 + *number of RDAREAs for index* × 4 + *number of index segments*

Also add the following if there are routines that reference the target table:

+ *number of routines that reference the target table*

Also add the following if there are routines that become invalid:

+ *number of routines that become invalid*

(i) HiRDB/Parallel Server (reference-only back-end server)

2

(6) ALTER TRIGGER

(a) For HiRDB/Single Server

19

Add the following if the executer is not the owner:

+ 2

Add the following if there is a trigger event column:

+ *number of trigger event columns* × 2 + 1

Add the following if a trigger action condition is specified:

+ *number of resource information items used in trigger action conditions* × 2 + 1

Add the following if there are resources used by the trigger:

+ *number of resources used* × 2 + 1

Add the following if there are tables in the resources used:

+ *number of tables used*

Add the following if there are view tables in the resources used:

+ *number of view tables used* + *number of utilized resources used in view tables used* + 1

Add the following if there are indexes in the resources to be used:

+ *number of indexes to be used* + 1

Add the following if there are routines in the resources to be used:

+ *number of routines used*

Add the following if there are user-defined types in the resources to be used:

+ *number of user-defined types to be used* + 1

Add the following if parameters are specified:

+ *number of parameters* × 2 + 1

(b) For HiRDB/Parallel Server (front-end server)

2

(c) For HiRDB/Parallel Server (dictionary server)

17

Add the following if the executer is not the owner:
+ 2

Add the following if there is a trigger event column:
+ *number of trigger event columns* × 2 + 1

Add the following if a trigger action condition is specified:
+ *number of resource information items used in trigger action conditions* × 2 + 1

Add the following if there are resources used by the trigger:
+ *number of resources used* × 2 + 1

Add the following if there are tables in the resources used:
+ *number of tables used*

Add the following if there are view tables in the resources used:
+ *number of view tables used* + *number of utilized resources used in view tables used* + 1

Add the following if there are indexes in the resources to be used:
+ *number of indexes to be used* + 1

Add the following if there are routines in the resources to be used:
+ *number of routines used*

Add the following if there are user-defined types in the resources to be used:
+ *number of user-defined types to be used* + 1

Add the following if parameters are specified:
+ *number of parameters* × 2 + 1

(d) For HiRDB/Parallel Server (back-end server)

1

(7) COMMENT

(a) For HiRDB/Single Server

3

Add the following if the comment is a column:
+ 2

(b) For HiRDB/Parallel Server (dictionary server)

3

Add the following if the comment is a column:
+ 2

(8) CREATE AUDIT

(a) For HiRDB/Single Server

7

Add the following if objects are specified:
+ *number of definitions of the objects in question*

Add the following if objects are not specified:

+ *number of definitions with no object specified*

(b) For HiRDB/Parallel Server (dictionary server)

7

Add the following if objects are specified:

+ *number of definitions of the objects in question*

Add the following if objects are not specified:

+ *number of definitions with no object specified*

(9) CREATE CONNECTION SECURITY

(a) For HiRDB/Single Server

7 + *number-of-users-registered-in-dictionary-table-SQL_USERS*

(b) For HiRDB/Parallel Server (front-end server)

1

(c) For HiRDB/Parallel Server (dictionary server)

6 + *number-of-users-registered-in-dictionary-table-SQL_USERS*

(10) CREATE [PUBLIC] FUNCTION

(a) For HiRDB/Single Server

number of locked resources in `CREATE [PUBLIC]PROCEDURE` + *number of routines whose objects become invalid*

Add the following if there are parameters:

+ *number of parameters* × 2

Add the following if a parameter includes a user-defined type:

+ *number of user-defined types* × 2

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 4 + 2 + *number of resources of routines that become invalid* × 5 + 1

Also add the following if a routine that becomes invalid is used by a view table:

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

(b) For HiRDB/Parallel Server (front-end and back-end server)

number of locked resources in `CREATE [PUBLIC]PROCEDURE` + *number of routines whose objects become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

number of locked resources in `CREATE [PUBLIC]PROCEDURE` + *number of routines whose objects become invalid*

Add the following if there are parameters:

+ *number of parameters* × 2

Add the following if a parameter includes a user-defined type:

+ *number of user-defined types* × 2

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 4 + 2 + *number of resources of routines that become invalid* × 5 + 1

Also add the following if a routine that becomes invalid is used by a view table:

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

(11) CREATE INDEX (not a plug-in index)

(a) For HiRDB/Single Server

20 + *number of routines in which indexes become invalid* + *number of member columns* × 4

Add the following if the indexes are not temporary table indexes:

+ 5 + 2 × *number of RDAREAs for index*

+ *number of RDAREAs for index* × 4

Add the following if there are routines that reference the table that defines the index:

+ *number of routines that reference the table that defines the index* × 4 + 2

Add the following if there are trigger action procedures that reference the table that defines the index:

+ *number of trigger action procedures that reference the table that defines the index* × 5

Add the following if there is more than one RDAREA for the index:

+ *number of RDAREAs for the index* × 3 + 1

Add the following if the index is a substructure index:

+ *number of index member substructure paths* × 3

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of routines in which the index becomes invalid*

(c) For HiRDB/Parallel Server (dictionary server)

19 + *number of member columns* × 4

Add the following if the indexes are not temporary table indexes:

+ *number of RDAREAs for index* × 4

Add the following if there are routines that reference the table that defines the index:

+ *number of routines that reference the table that defines the index* × 4 + 2

Add the following if there are trigger action procedures that reference the table that defines the index:

+ *number of trigger action procedures that reference the table that defines the index* × 5

Add the following if there is more than one RDAREA for the index:

+ *number of RDAREAs for the index* × 3 + 1 + *number of routines in which the index becomes invalid*

Add the following if the index is a substructure index:

+ *number of index member substructure paths* × 3

(d) For HiRDB/Parallel Server (back-end server) (non-temporary table indexes)

5 + *number of RDAREAs for the index* × 2 + *number of routines in which the index becomes invalid*

(e) For HiRDB/Parallel Server (back-end server) (temporary table indexes)

number of routines in which indexes become invalid

(f) For HiRDB/Parallel Server (reference-only back-end server)

2

(12) CREATE INDEX (plug-in index)

(a) For HiRDB/Single Server

21 + *number of routines in which indexes become invalid* + *number of RDAREAs for index* × 8 + *number of member columns* × 4 + *number of functions applied* × 5 + Σ *number of parameters of the functions applied*

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of routines in which indexes become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

15 + *number of RDAREAs for index* × 6 + *number of member columns* × 4 + *number of functions applied* × 5 + Σ *number of parameters of the functions applied* + *number of routines in which indexes become invalid*

(d) For HiRDB/Parallel Server (back-end server)

5 + *number of RDAREAs for index* × 2 + *number of routines in which indexes become invalid*

(13) CREATE [PUBLIC] PROCEDURE

(a) For HiRDB/Single Server

13 + *number of tables accessed in SQL statement preprocessing* + *number of view tables accessed in SQL statement preprocessing* + *number of base tables that are the basis of view tables accessed in SQL statement preprocessing*

Add the following if a resource to be used exists:

+ *number of resources to be used* × 7 + 3

Add the following if parameters are specified:

+ *number of parameters* × 3 + 1

Add the following if the parameter data type is a user-defined type:

+ *number of user-defined type parameters* × 2 + 1

(b) For HiRDB/Parallel Server (front-end server)

3 + *number of tables accessed in SQL statement preprocessing* + *number of view tables accessed in SELECT statement preprocessing* + *number of base tables that are the basis of view tables accessed in SELECT statement preprocessing*

(c) For HiRDB/Parallel Server (dictionary server)

15

Add the following if a resource to be used exists:

+ *number of resources to be used* × 7 + 3

Add the following if parameters are specified:

+ *number of parameters* × 3 + 1

Add the following if the parameter data type is a user-defined type:

+ *number of user-defined type parameters* × 2 + 1

(14) CREATE SCHEMA

(a) For HiRDB/Single Server

5

(b) For HiRDB/Parallel Server (dictionary server)

5

(15) CREATE SEQUENCE

(a) For HiRDB/Single Server

9 + 2

(b) For HiRDB/Parallel Server (dictionary server)

9

(c) For HiRDB/Parallel Server (back-end server)

2

(16) CREATE TABLE

(a) For HiRDB/Single Server (RDAREA for table is omitted and the table is not row-partitioned (including temporary tables))

20 + *number of columns* × 3

Add the following if the table is not a temporary table:

+ 9
+ *number of public user RDAREAs* × 3
+ *number of LOB columns* × 4

Add the following if a user-defined type is used:

+ *number of user-defined types* × 6 + *number of LOB attributes* × *number of partitioned RDAREAs* × 4 + *number of user-defined attributes* × 2 + 5

Add the following if a cluster key is defined:

+ 8
+ *number of member columns* × 3

Add the following if the table is not a temporary table:

+ 5
+ *number of RDAREAs for index* × 2
+ *number of partitioned RDAREAs* × 6

Add the following if a referential constraint is defined:

+ *number of referenced tables*
+ *number of foreign keys* × 10

Add the following if CASCADE is defined for referential constraint operation:

+*number of CASCADEs* × 17
+*number of referenced tables specified by CASCADE* × 4

Add the following if the specification of referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of the referenced table for which CASCADE is specified* × 8
+ *number of referenced tables specified by CASCADE* × 3

Add the following if there are functions that become invalid:

+ *number of functions whose objects are invalid*
+ Σ (2 + *number of resources inside procedure* × 5)
+ *total number of functions that reference the referenced tables*

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

(b) For HiRDB/Single Server (RDAREA for table is specified and the table is not row-partitioned)

28 + *number of columns* × 4 + *number of LOB columns* × 4 + *number of RDAREAs for table* × 4

Add the following if a user-defined type is used:

+ *number of user-defined types* × 6

+ *number of LOB attributes* × *number of partitioned RDAREAs* × 4

+ *number of user-defined type attributes* × 2

+ 5

Add the following if a cluster key is defined:

+ 5 + *number of RDAREAs for index* × 2 + *number of partitioned RDAREAs* × 6 + 8 + *number of member columns* × 3

Add the following if a referential constraint is defined:

+ *number of referenced tables*

+ *number of foreign keys* × 10

Add the following if CASCADE is defined for referential constraint operation:

+*number of CASCADEs* × 17

+*number of referenced tables specified by CASCADE* × 4

Add the following if the specification of referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of the referenced table for which CASCADE is specified* × 8

+ *number of referenced tables specified by CASCADE* × 3

Add the following if there are functions that become invalid:

+ *number of functions whose objects are invalid*

+ Σ (2 + *number of resources inside procedure* × 5)

+ *total number of functions that reference the referenced tables*

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

(c) For HiRDB/Single Server (row-partitioned table)

29 + *number of columns* × 4 + *number of partitioned RDAREAs* × 21 + *number of LOB columns* × *number of partitioned RDAREAs* × 4

Add the following if a user-defined type is used:

+ *number of user-defined types* × 6

+ *number of LOB attributes* × *number of partitioned RDAREAs* × 4

+ *number of user-defined type attributes* × 2

+ 5

Add the following if a cluster key is defined:

+ 5 + *number of RDAREAs for index* × 2 + *number of partitioned RDAREAs* × 6 + 8 + *number of member columns* × 3

Add the following if a referential constraint is defined:

+ *number of referenced tables*

+ *number of foreign keys* × 10

Add the following if CASCADE is defined for referential constraint operation:

+*number of CASCADEs* × 17

+*number of referenced tables specified by CASCADE* × 4

Add the following if the specification of referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of the referenced table for which CASCADE is specified* × 8

+ *number of referenced tables specified by CASCADE* × 3

Add the following if there are functions that become invalid:

+ *number of functions whose objects are invalid*

+ Σ (2 + *number of resources inside procedure* × 5)

+ *total number of functions that reference the referenced tables*

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

(d) For HiRDB/Parallel Server (front-end server)

3

Add the following if a referential constraint is defined:

+ *number of referenced tables*

+ *total number of routines that reference the referenced tables*

(e) For HiRDB/Parallel Server (dictionary server) (RDAREA for table is omitted and the table is not row-partitioned (including temporary tables))

20 + *number of columns* × 3

Add the following if the table is not a temporary table:

+ *number of public user RDAREAs* × 3

+ *number of LOB columns* × *number of partitioned RDAREAs* × 4

Add the following if user-defined types are used:

+ *number of user-defined types* × 6

+ *number of LOB attributes* × *number of partitioned RDAREAs* × 4

+ *number of user-defined type attributes* × 2

+ 5

Add the following if cluster keys are defined:

+ 8 + *number of member columns* × 3

Add the following if the table is not a temporary table:

+ *number of partitioned RDAREAs* × 6

Add the following if referential constraint is defined:

+ *number of foreign keys* × 10

Add the following if CASCADE is defined for the referential constraint operation:

+ *number of CASCADEs* × 17

+ *number of referenced tables specified by CASCADE* × 4

Add the following if the specification of the referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of the referenced table for which CASCADE is specified* × 8

+ *number of referenced tables specified by CASCADE* × 3

Add the following if there are functions that become invalid:

+ *number of functions whose objects are invalid*

+ Σ (2 + *number of resources inside procedure* × 5)

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

(f) For HiRDB/Parallel Server (dictionary server, RDAREA for table is specified and the table is not row-partitioned)

25 + *number of columns* × 4 + *number of LOB columns* × 4

Add the following if a user-defined type is used:

+ *number of user-defined types* × 6

+ *number of LOB attributes* × *number of partitioned RDAREAs* × 4

+ *number of user-defined type attributes* × 2

+ 5

Add the following if a cluster key is defined:

+ *number of partitioned RDAREAs* × 6 + 8 + *number of member columns* × 3

Add the following if a referential constraint is defined:

+ *number of foreign keys* × 10

Add the following if CASCADE is defined for referential constraint operation:

+*number of CASCADEs* × 17

+*number of referenced tables specified by CASCADE* × 4

Add the following if the specification of referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of the referenced table for which CASCADE is specified* × 8

+ *number of referenced tables specified by CASCADE* × 3

Add the following if there are functions that become invalid:

+ *number of functions whose objects are invalid*

+ Σ (2 + *number of resources inside procedure* × 5)

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

(g) For HiRDB/Parallel Server (dictionary server, row-partitioned table)

Add the following if the table is not a matrix-partitioned table:

24 + *number of columns* × 4 + *number of partitioned RDAREAs* × 17 + *number of LOB columns* × *number of partitioned RDAREAs* × 4

Add the following if the table is a matrix-partitioned table:

21 + *number of columns* × 4 + *number of partitioned RDAREAs* (including overlapped ones) × 16 + *number of LOB columns* × *number of partitioned RDAREAs* × 4 + *number of partitioning keys* × 2 + (*number of storage conditions* + 2) × 2 + 2

Add the following if a user-defined type is used:

+ *number of user-defined types* × 6 + *number of LOB attributes* × *number of partitioned RDAREAs* × 4 + *number of user-defined attributes* × 2 + 5

Add the following if a cluster key is defined:

+ *number of partitioned RDAREAs* × 6 + 8 + *number of member columns* × 3

Add the following if a referential constraint is defined:

+ *number of foreign keys* × 10

Add the following if CASCADE is defined for referential constraint operation:

+*number of CASCADEs* × 17

+*number of referenced tables specified by CASCADE* × 4

Add the following if the specification of referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of the referenced table for which CASCADE is specified* × 8

+ *number of referenced tables specified by CASCADE* × 3

Add the following if there are functions that become invalid:

+ *number of functions whose objects are invalid*

+ Σ (2 + *number of resources inside procedure* × 5)

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

(h) For HiRDB/Parallel Server (back-end server) (non-temporary table)

5 + *number of RDAREAs for table* × 4

Add the following if a cluster key is defined:

+ 5 + *number of RDAREAs for index* × 2

(i) For HiRDB/Parallel Server (back-end server) (temporary table)

0

(j) For HiRDB/Parallel Server (reference-only back-end server)

2

Add the following if a cluster key is defined:

+ 1

(17) CREATE TRIGGER

(a) For HiRDB/Single Server

21 + *number of routines in which objects become invalid* × 5 + *number of resources used by trigger action procedures* × 5

Add the following if a trigger action condition is specified:

+ *number of resources used inside trigger action condition specification* × 5 + 3

Add the following if a column name modified by an old or new values correlation name is used inside a trigger action procedure:

+ *number of column types modified by new values correlation names* × 3 + *number of column types modified by old values correlation names* × 3 + 3

Add the following if trigger event columns are specified:

+ *number of trigger event columns* × 3

Add the following if there are routines that become invalid:

+ *number of invalid routines* × 5 + *number of resources used by routines* × 5

Add the following if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(b) For HiRDB/Parallel Server (front-end server)

3 + *number of routines in which objects become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

18 + *number of routines in which objects become invalid* + *number of resources used by trigger action procedures* × 5

Add the following if a trigger action condition is specified:

+ *number of resources used inside trigger action condition specification* × 5 + 3

Add the following if a column name modified by an old or new values correlation name is used inside a trigger action procedure:

+ *number of column types modified by new values correlation names* × 3 + *number of column types modified by old values correlation names* × 3 + 3

Add the following if trigger event columns are specified:

+ *number of trigger event columns* × 3

Add the following if there are routines that become invalid:

+ *number of invalid routines* × 5 + *number of resources used by routines* × 5

Add the following if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(d) For HiRDB/Parallel Server (back-end server)

number of routines in which objects become invalid

(18) CREATE TYPE

(a) For HiRDB/Single Server

22 + *number of routines that become invalid* + *number of attributes* × 3

Add the following if there are user-defined types in the attributes:

+ (*number of user-defined types* × 6 + 2)

Add the following for subtype definitions:

+ 11

Add the following if there are routines that become invalid:

+ (*number of routines that become invalid* × 4 + 2 + *number of resources of the routines that become invalid* × 5 + 1)

Also add the following if a routine that becomes invalid is used by a view table:

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

Add the following if a function or procedure is defined:

+ *number of locked resources for function/procedure definitions*#

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of routines that become invalid*

Add the following if a function or procedure is defined:

+ *number of locked resources for function/procedure definitions* #

(c) For HiRDB/Parallel Server (dictionary server)

21 + *number of attributes* × 3

Add the following if there are user-defined types in the attributes:

+ (*number of user-defined types* × 6 + 2)

Add the following for subtype definitions:

+ 11

Add the following if there are routines that become invalid:

+ (*number of routines that become invalid* × 4 + 2 + *number of resources of the routines that become invalid* × 5 + 1)

Also add the following if a routine that becomes invalid is used by a view table:

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

Add the following if a function or procedure is defined:

+ *number of locked resources for function/procedure definitions* #

(d) For HiRDB/Parallel Server (back-end server)

number of routines that become invalid

#: Reference the lock request count of the function definition or the procedure definition.

(19) CREATE [PUBLIC] VIEW

(a) For HiRDB/Single Server

31 + *number of tables accessed in SELECT statement preprocessing* + *number of view tables accessed in SELECT statement preprocessing* + *number of base tables that are the basis of view tables accessed in SELECT statement preprocessing* + *number of columns* × 4 + *number of tables that are the basis of the view table* × 4 + *number of SELECT statements preprocessing*

Add the following if the table is a public view table:

+ 2

+ *number of tables having the same name as the public view table* × 2

+ *number of view tables having the same name as the public view table* × 2

(b) For HiRDB/Parallel Server (front-end server)

3 + *number of tables accessed in SELECT statement preprocessing* + *number of view tables accessed in SELECT statement preprocessing* + *number of base tables that are the basis of view tables accessed in SELECT statements preprocessing*

(c) For HiRDB/Parallel Server (dictionary server)

28 + *number of columns* × 4 + *number of tables that are the basis of the view table* × 4 + *number of SELECT statements preprocessing*

Add the following if the table is a public view table:

+ 2
+ *number of tables having the same name as the public view table* × 2
+ *number of view tables having the same name as the public view table* × 2

(20) DEALLOCATE MEMORY TABLE

For `DEALLOCATE MEMORY TABLE`, determine the value if you will be using a memory-resident database.

(a) For HiRDB/Single Server

4 + *number of indexes defined for the table*

(b) For HiRDB/Parallel Server (dictionary server)

4 + *number of indexes defined for the table*

(21) DROP AUDIT

(a) For HiRDB/Single Server

7

Add the following if objects are specified:

+ *number of definitions of the objects in question*

Add the following if objects are not specified:

+ *number of definitions with no object specified*

(b) For HiRDB/Parallel Server (dictionary server)

7

Add the following if objects are specified:

+ *number of definitions of the objects in question*

Add the following if objects are not specified:

+ *number of definitions with no object specified*

(22) DROP CONNECTION SECURITY

(a) For HiRDB/Single Server

7 + *number-of-users-registered-in-dictionary-table-SQL_USERS*

(b) For HiRDB/Parallel Server (front-end server)

1

(c) For HiRDB/Parallel Server (dictionary server)

6 + *number-of-users-registered-in-dictionary-table-SQL_USERS*

(23) DROP DATA TYPE

(a) For HiRDB/Single Server

26 + *number of routines that become invalid* + *number of attributes* × 3

Add the following for a subtype:

+ 6

Add the following if there are user-defined type attributes:

+ *number of user-defined type attributes* × 4

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 4 + 2 + *number of resources of the routines that become invalid* × 5 + 1

Also add the following if a routine that becomes invalid is used by a view table:

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

Add the following if there are function definitions:

+ *number of function definitions* × 7 + 1 + *number of function parameters* × 4 + 1 + *number of procedures used by the functions* + 1

Add the following if there are procedure definitions:

+ *number of procedure definitions* × 5 + 1 + *number of procedure parameters* × 4 + 1 + *number of resources used by procedures* × 5 + 1

Add the following if a trigger action condition is specified:

+ 15

Add the following if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of routines that become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

23 + *number of attributes* × 3 + *number of routines whose objects become invalid*

Add the following for a subtype:

+ 6

Add the following if there are user-defined type attributes:

+ *number of user-defined type attributes* × 4

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 4 + 2 + *number of resources of the routines that become invalid* × 5 + 1

Also add the following if a routine that becomes invalid is used by a view table:

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

Add the following if there are function definitions:

+ *number of function definitions* × 7 + 1 + *number of function parameters* × 4 + 1 + *number of procedures used by the functions* × 5 + 1

Add the following if there are procedure definitions:

+ *number of procedure definitions* × 5 + 1 + *number of procedure parameters* × 4 + 1 + *number of resources used by procedures* × 5 + 1

Add the following if a trigger action condition is specified:

+ 15

Add the following if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(d) For HiRDB/Parallel Server (back-end server)

number of routines whose objects become invalid

(24) DROP [PUBLIC] FUNCTION

(a) For HiRDB/Single Server

Number of `DROP [PUBLIC ]PROCEDURE` locked resources

Add the following if there are parameters:

+ *number of parameters* × 2

Add the following if a parameter includes a user-defined type:

+ *number of user-defined types* × 2

Add the following if a trigger action condition is specified:

+ 15

Add the following if a routine that becomes invalid is used by a view table.

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

(b) For HiRDB/Parallel Server (front-end and back-end servers)

same as DROP [PUBLIC]PROCEDURE

(c) For HiRDB/Parallel Server (dictionary server)

number of locked resources in DROP [PUBLIC]PROCEDURE

Add the following if there are parameters:

+ *number of parameters* × 2

Add the following if a parameter includes a user-defined type:

+ *number of user-defined types* × 2

Add the following if a trigger action condition is specified:

+ 15

Add the following if a routine that becomes invalid is used by a view table.

+ *number of routines used by view tables that become invalid* + 1 + *number of routines used by view tables that become invalid* × *number of parameters* + 1

(25) DROP INDEX (not a plug-in index)

(a) For HiRDB/Single Server

22 + *number of routines in which objects or indexes become invalid* + *number of member columns* × 3

Add the following if the indexes are not temporary table indexes:

+ *number of RDAREAs for index* × 7

+ *number of index segments*

Add the following if the indexes are temporary table indexes:

+ 3

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 7 + 1

Add the following if there are trigger action procedures that reference the table that defines the index:

+ *number of trigger action procedures that reference the table that defines the index* × 5

Add the following if acquiring index optimization information:

+ 4

Add the following if the index is a substructure index:

+ *number of index member substructure paths* × 3

Add the following if using the facility for predicting reorganization time:

+ *number of RDAREAs for the index* × 62 + 1

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of routines in which objects or the index become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

16 + *number of member columns* × 3 + *number of routines in which objects or indexes become invalid*

Add the following if the indexes are not temporary table indexes:

+ *number of RDAREAs for index* × 3

Add the following if there are routines that become invalid:

number of routines that become invalid × 7 + 1

Add the following if there are trigger action procedures that reference the table that defines the index:

+ *number of trigger action procedures that reference the table that defines the index* × 5

Add the following if acquiring index optimization information:

+ 4

Add the following if the index is a substructure index:

+ *number of index member substructure paths* × 3

Add the following if using the facility for predicting reorganization time:

+ *number of RDAREAs for the index* × 62 + 1

(d) For HiRDB/Parallel Server (back-end server) (non-temporary table indexes)

5 + *number of RDAREAs for the index* × 4 + *number of index segments* + *number of routines in which objects or the index become invalid*

(e) For HiRDB/Parallel Server (back-end server) (temporary table indexes)

number of routines in which objects or indexes become invalid

(f) For HiRDB/Parallel Server (reference-only back-end server)

2

(26) DROP INDEX (plug-in index)

(a) For HiRDB/Single Server

15 + *number of routines in which objects or indexes become invalid* + *number of RDAREAs storing the index* × 10 + *number of member columns* × 4 + *number of applicable functions* × 3 + *number of index segments*

Add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs for index* × 62 + 1

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of routines in which objects or indexes become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

9 + *number of RDAREAs storing the index* × 6 + *number of member columns* × 4 + *number of applicable functions* × 3 + *number of routines in which objects or indexes become invalid*

Add the following if the facility for predicting reorganization time is used:

+ *number of RDAREAs for index* × 62 + 1

(d) For HiRDB/Parallel Server (back-end server)

5 + *number of RDAREAs storing the index* × 4 + *number of index segments* + *number of routines in which objects or indexes become invalid*

(27) DROP [PUBLIC] PROCEDURE

(a) For HiRDB/Single Server#

19 + *number of parameters* × 5 + *number of resources* × 10 + *number of routines in which objects become invalid*

Add the following value if there are routines that become invalid:

+ *number of routines that become invalid* × 4 + 2 + *number of resources for the routines that become invalid* × 5 + 1

Add the following value if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of routines in which objects become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

18 + *number of parameters* × 3 + *number of resources* × 10 + *number of routines in which objects become invalid*

Add the following value if there are routines that become invalid:

+ *number of routines that become invalid* × 4 + 2 + *number of resources for the routines that become invalid* × 5 + 1

Add the following if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(d) For HiRDB/Parallel Server (back-end server)

number of routines in which objects become invalid

(28) DROP SCHEMA

(a) For HiRDB/Single Server

11 + *number of tables inside the schema* + *number of view tables inside the schema* + *number of routines inside the schema* + *number of view tables in other schemas that use tables or view tables inside the schema as base tables* + *number of indexes in other schemas that reference the tables, view tables, routines, or data types to be deleted*

Add the number of tables inside the schema that are to be deleted (when they are not matrix-partitioned tables).

+ Σ {*number of RDAREAs storing the tables* × 3 + *number of specified partitioning conditions* × 9 + *number of columns* × 4 + (*number of LOB columns* × 4) × *number of RDAREAs storing the tables* + 1 + 8} + 2

+ Σ (*number of RDAREAs for table* × 10

+ 10

+ *number of table data segments being used*)

Add the number of tables inside the schema that are to be deleted (when they are matrix-partitioned tables).

+ Σ {*number of RDAREAs storing the tables* × 3 + *number of specified RDAREAs storing the tables (including overlapped ones)* × 12 + *number of columns* × 4 + (*number of LOB columns* × 4) × *number of RDAREAs storing the tables* + 1 + 8 + *number of privilege definitions for the tables to be deleted* + *number of partitioning keys* × 2 + (*number of specified storage conditions* + 2) × 2 + 2} + 3

+ Σ (*number of RDAREAs for table* × 10

+ 10

+ *number of table data segments being used*)

Add the number of indexes inside the schema that are to be deleted.

+ 3 + Σ (*number of RDAREAs for indexes* × 7 + *number of member columns* × 3 + *number of index member substructure paths* × 3#1) + Σ (*number of RDAREAs for the indexes* × 8 + 10 + *number of index segments being used*) + 8

Add the following if optimization information is collected:

+ *number of tables for which optimization information is collected* × 2 + *number of columns for which optimization information is collected* × 3 + *number of indexes for which optimization information is collected* × 3

Add the number of view tables that are deleted along with the tables being deleted:

+ 2 + Σ (12 + *number of base tables for view tables* × 4 + *number of columns* × 4)

Add the number of routines that use tables, view tables, routines, or data types inside the schema:

+ 3 + Σ (5 + *number of resources in procedures* × 5 + *number of defined parameters* × 3)

Add the following if there are procedures that become invalid:

+ *number of invalid procedures* + Σ (2 + *number of resources* × 5) + *number of data types defined inside the schema*#2

Add the following if there are trigger action procedures that reference tables to be deleted:

+ *number of trigger action procedures that reference tables to be deleted* × 5

Add the following if there are tables for which triggers are defined:

+ *number of triggers defined inside the schema* × 8 + *total number of columns specified in the UPDATE trigger definition inside the schema* × 3 + *number of triggers that have trigger action conditions inside the schema* × 3 + *number of resource types referenced in the trigger action conditions inside the schema* × 5

Add the following if a referential constraint is defined:

+ *number of referenced tables* + *number of foreign keys* × 10

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

Add the following if using the facility for predicting reorganization time:

+ Σ {(*number of RDAREAs storing the tables* + (*number of LOB columns* + *number of LOB attributes*) × *number of table-storage RDAREAs*) × 62} + Σ {(*number of RDAREAs for indexes* + (*number of LOB columns* + *number of LOB attributes*) × *number of table-storage RDAREAs*) × 62} + 1

Add the following if there are sequence generators inside the schema:

+ 2 + *number of sequence generators inside the schema* × 3 + *number of sequence generators inside the schema* × 2

(b) For HiRDB/Parallel Server (front-end server)

3 + *number of tables inside the schema* + *number of view tables inside the schema* + *number of routines inside the schema* + *number of sequence generators inside the schema* + *number of view tables in other schemas that use tables or view tables inside the schema as base tables* + *number of routines in other schemas that reference the tables, view tables, routines, or data types to be deleted*

(c) For HiRDB/Parallel Server (dictionary server)

11 + *number of indexes in other schemas*

Add the number of tables inside the schema that are to be deleted (when they are not matrix-partitioned tables).

+ Σ {*number of RDAREAs storing the tables* × 3 + *number of specified partitioning conditions* × 9 + *number of columns* × 4 + (*number of LOB columns* × 4) × *number of RDAREAs storing the tables* + *number of privilege definitions for the tables to be deleted* × 3 + 6} + 11

Add the number of tables inside the schema that are to be deleted (when they are matrix-partitioned tables).

+ Σ {*number of RDAREAs storing the tables* × 3 + *number of specified RDAREAs storing the tables (including overlapped ones)* × 12 + *number of columns* × 4 + (*number of LOB columns* × 4) × *number of RDAREAs storing the tables* + *number of privilege definitions for the tables to be deleted* × 3 + 1 + 6 + *number of partitioning keys* × 2 + (*number of specified storage conditions* + 2) × 2 + 2} + 3

Add the number of indexes inside the schema that are to be deleted.

+ 4 + Σ (*number of RDAREAs for the indexes* × 7 + *number of member columns* × 7 + *number of index member substructure paths* × $3^{\#1}$) + *number of indexes* × 5

Add the following if optimization information is collected:

+ *number of tables for which optimization information is collected* × 2 + *number of columns for which table optimization information is collected* × 3 + *number of indexes for which optimization information is collected* × 3

Add the number of view tables that are deleted along with the tables being deleted:

+ 2 + Σ (18 + *number of base tables for the view tables* × 4 + *number of columns* × 4 + *number of privilege definitions for the view tables* × 3)

Add the number of routines that use tables, view tables, routines, or data types inside the schema:

+ 3 + Σ (5 + *number of resources inside the procedure* × 10 + *number of defined parameters* × 3)

Add the following if there are procedures that become invalid:

+ *number of invalid procedures* + Σ (5 + *number of resources* × 5) + *number of for data types defined inside the schema*$^{\#2}$

Add the following if there are trigger action procedures that reference tables to be deleted:

+ *number of trigger action procedures that reference tables to be deleted* × 5

Add the following if there are tables for which triggers are defined:

+ *number of triggers defined inside the schema* × 8 + *total number of columns specified in the UPDATE trigger definition inside the schema* × 3 + *number of triggers that have trigger action conditions inside the schema* × 3 + *number of resource types referenced in the trigger action conditions inside the schema* × 5

Add the following if a referential constraint is defined:

+ *number of referenced tables* + *number of foreign keys* × 10

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

Add the following if using the facility for predicting reorganization time:

+ Σ ((*number of RDAREAs storing the tables* + (*number of LOB columns* + *number of LOB attributes*) × *number of table-storage RDAREAs*) × 62) + Σ ((*number of RDAREAs for indexes* + (*number of LOB columns* + *number of LOB attributes*) × *number of table-storage RDAREAs*) × 62) + 1

Add the following if there are sequence generators inside the schema:

+ 2 + *number of sequence generators inside the schema* × 2

(d) For HiRDB/Parallel Server (back-end server)

8 + *number of routines whose objects or index objects become invalid*

Add the following for non-temporary tables or non-temporary table indexes:

+ Σ (*number of RDAREAs storing tables* × 10 + 10 + *number of table data segments being used*)

+ Σ (*number of RDAREAs for indexes* × 8 + 10 + *number of index segments being used*)

Add the following if there is a sequence generator inside the schema:

+ *number of sequence generators inside the schema* × 2

(e) For HiRDB/Parallel Server (reference-only back-end server)

number of tables + *number of RDAREAs for indexes*

#1: Add iteratively if the index is a substructure index.

#2: See the number of lock requests for user-defined deletion.

(29) DROP SEQUENCE

(a) For HiRDB/Single Server

9 + 2

Add the following if there are routines that become invalid:

+ *number of routines whose objects become invalid* × 3 + 2

Add the following if triggers are defined:

+ *number of trigger definitions* × 2 + 2 + *number of defined trigger event columns* + 1 + *number of parameters used by defined triggers* + 1

(b) For HiRDB/Parallel Server (front-end server)

1

Add the following if there are routines that become invalid:

+ *number of routines whose objects become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

8

Add the following if there are routines that become invalid:

+ *number of routines whose objects become invalid* × 2 + 2

Add the following if triggers are defined:

+ *number of trigger definitions* × 2 + 2 + *number of defined trigger event columns* + 1 + *number of parameters used by defined triggers* + 1

(d) For HiRDB/Parallel Server (back-end server)

2

(30) DROP TABLE

(a) For HiRDB/Single Server

Add the following if the table is not a matrix-partitioned table:

24
+ *number of view tables defined based on the table*
+ *number of routines that reference the table or view table to be deleted*
+ *number of columns* × 4

Add the following if the table is not a temporary table:

+ 5
+ *number of RDAREAs for table* × 4
+ *number of table data segments being used*
+ *number of RDAREAs for table* × 3
+ *number of specified partitioning conditions* × 9

Add the following if the table is a matrix-partitioned table:

22 + *number of view tables defined based on the table* + *number of routines that reference the table or view tables to be deleted* + *number of RDAREAs storing the table* × 3 + *number of RDAREAs storing the table (including overlapped ones)* × 8 + *number of columns* × 4 + 5 + *number of RDAREAs storing the table* × 4 + *number of table data segments being used* + *number of partitioning keys* × 2 + (*number of specified storage conditions* + 2) × 2 + 2

Add the following if indexes are defined:

+ Σ (*number of member columns* × 3) + 3

Add the following if the indexes are not temporary table indexes:

+ Σ (*number of RDAREAs for index* × 2)
+ *number of index segments being used*
+ Σ {(*number of RDAREAs for index* × 7) + (*number of paths making up the index component* × 3)#1}

Add the following if LOB columns are defined:

+ (*number of LOB columns* × 4) × *number of RDAREAs storing the table* + 1

Add the following if optimization information is collected:

+ 2 + *number of columns for which optimization information is collected* × 3 + *number of indexes for which optimization information is collected* × 3

Add the following for the base tables of view tables:

+ *number of view tables to be deleted* + Σ {(12 + *number of base tables for the view tables*) + *number of columns* × 4} + 2

Add the following if there are routines that become invalid:

+ *number of routines whose objects become invalid* + Σ (2 + *number of resources inside the procedure* × 5)

Add the following if there are trigger action procedures that reference the table to be deleted:

+ *number of trigger action procedures that reference the table to be deleted* × 5

Add the following if user-defined type columns are defined:

+ *number of user-defined type columns* × 4 + *number of LOB attributes* × *number of partitioned RDAREAs* × 5 + *number of abstract data-type attributes* + 4

Add the following if triggers are defined:

+ *number of trigger definitions* × 8 + *total number of columns specified in the UPDATE trigger definition* × 3 + *number of triggers that have trigger action conditions* × 3 + *number of resource types referenced in the trigger action conditions* × 5

Add the following if a user-defined type that includes a LOB column or LOB attribute is defined:

+ Σ (↑ *number of segments allocated to RDAREAs* ÷ 64,000 ↑ + 1)#2

Add the following if a referential constraint is defined:

+ *number of referenced tables* + *number of foreign keys* × 10 + *total number of functions in referenced tables that are affected by deletion of constraints*

Also add the following if CASCADE is defined for the referential constraint operation:

+ *number of CASCADEs* × 17 + *number of referenced tables specified by CASCADE* × 4

Also add the following if the specification of the referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of a referenced table for which CASCADE is specified* × 8 + *number of referenced tables specified by CASCADE* × 3

Add the following if a check constraint is defined:

+ *number of check constraints* × 9

Add the following if using the facility for predicting reorganization time:

+ {*number of table-storage RDAREAs* + *number of RDAREAs for indexes* + (*number of LOB columns* + *number of LOB attributes*) × *number of table-storage RDAREAs*} × 62 + 1

(b) For HiRDB/Parallel Server (front-end server)

3 + *number of view tables defined based on the table* + *number of routines that reference the table or view tables to be deleted*

Add the following if a referential constraint is defined:

+ *number of referenced tables* × 4 + *total number of functions in referenced tables that are affected by deletion of constraints*

(c) For HiRDB/Parallel Server (dictionary server)

Add the following if the table is not a matrix-partitioned table:

21 + *number of columns* × 4

+ *number of routines in which objects become invalid*

Add the following if the table is not a temporary table:

+ *number of RDAREAs for table* × 3

+ *number of specified partitioning conditions* × 9

Add the following if the table is a matrix-partitioned table:

19 + *number of RDAREAs storing the table* × 3 + *number of specified partitioning conditions* × 9 + *number of columns* × 4 + *number of routines whose objects become invalid*

Add the following if indexes are defined:

+ Σ (*number of member columns* $\times$ 3) + 3

Add the following if the table is not a temporary table:

+ Σ {(*number of RDAREAs for index* $\times$ 7) + (*number of paths making up the index component* $\times$ $3^{\#1}$)}

Add the following if LOB columns are defined:

+ (*number of LOB columns* $\times$ 4) $\times$ *number of RDAREAs storing the table* + 1

Add the following if optimization information is collected:

+ 2 + *number of columns for which optimization information is collected* $\times$ 3 + *number of indexes for which optimization information is collected* $\times$ 3

Add the following for the base tables of view tables:

+ *number of view tables to be deleted* + Σ {12 + *number of base tables for the view tables*) + *number of columns* $\times$ 4} + 2

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* + Σ (2 + *number of resources inside the procedure* $\times$ 5)

Add the following if there are trigger action procedures that reference the table to be deleted:

+ *number of trigger action procedures that reference the table to be deleted* $\times$ 5

Add the following if user-defined type columns are defined:

+ *number of user-defined type columns* $\times$ 4 + *number of LOB attributes* $\times$ *number of partitioned RDAREAs* $\times$ 5 + *number of abstract data-type attributes* + 4

Add the following if triggers are defined:

+ *number of trigger definitions* $\times$ 8 + *total number of columns specified in the UPDATE trigger definition* $\times$ 3 + *number of triggers that have trigger action conditions* $\times$ 3 + *number of resource types referenced in the trigger action conditions* $\times$ 5

Add the following if a referential constraint is defined:

+ *number of referenced tables* + *number of foreign keys* $\times$ 10

Also add the following if CASCADE is defined for the referential constraint operation:

+ *number of CASCADEs* $\times$ 17 + *number of referenced tables specified by CASCADE* $\times$ 4

Also add the following if the specification of the referential constraint operation is UPDATE ON CASCADE:

+ *total number of primary key member columns of the referenced tables for which CASCADE is specified* $\times$ 8 + *number of referenced tables specified by CASCADE* $\times$ 3

Add the following if a check constraint is defined:

+ *number of check constraints* $\times$ 9

Add the following if using the facility for predicting reorganization time:

+ {*number of table-storage RDAREAs* + *number of RDAREAs for indexes* + (*number of LOB columns* + *number of LOB attributes*) $\times$ *number of table-storage RDAREAs*} $\times$ 62 + 1

(d) For HiRDB/Parallel Server (back-end server) (non-temporary table)

5 + *number of RDAREAs storing the table* $\times$ 4 + *number of table data segments being used* + *number of routines whose objects become invalid*

Add the following if indexes are defined:

+ Σ (*number of RDAREAs for indexes* $\times$ 2) + *number of index segments being used*

Add the following if a user-defined type that includes a LOB column or LOB attribute is defined:

+ Σ ($\uparrow$ *number of segments allocated to RDAREAs* $\div$ 64,000 $\uparrow$ + $1)^{\#2}$

(e) For HiRDB/Parallel Server (back-end server) (temporary table)

number of routines in which objects or indexes become invalid

(f) For HiRDB/Parallel Server (reference-only back-end server)

2

Add the following if shared indexes are defined:

+ *number of RDAREAs for indexes*

#1: Add iteratively if the index is a substructure index.

#2: Repeat and add this for the number of LOB RDAREAs.

(31) DROP TRIGGER

(a) For HiRDB/Single Server

19 + *number of routines in which objects become invalid* + *number of resources used by trigger action procedures* × 5

Add the following if a trigger action condition is specified:

+ *number of resources used inside trigger action condition specification* × 5 + 3

Add the following if a column name modified by an old or new values correlation name is used inside a trigger action procedure:

+ *number of column types modified by new values correlation names* × 3 + *number of column types modified by old values correlation names* × 3 + 3

Add the following if trigger event columns are specified:

+ *number of trigger event columns* × 3

Add the following if there are routines that become invalid:

+ *number of invalid routines* × 5 + *number of resources used by routines* × 5

Add the following if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(b) For HiRDB/Parallel Server (front-end server)

3 + *number of routines in which objects become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

16 + *number of routines in which objects become invalid* + *number of resources used by trigger action procedures* × 5

Add the following if a trigger action condition is specified:

+ *number of resources used inside trigger action condition specification* × 5 + 3

Add the following if a column name modified by an old or new values correlation name is used inside a trigger action procedure:

+ *number of column types modified by new values correlation names* × 3 + *number of column types modified by old values correlation names* × 3 + 3

Add the following if trigger event columns are specified:

+ *number of trigger event columns* × 3

Add the following if there are routines that become invalid:

+ *number of invalid routines* × 5 + *number of resources used by routines* × 5

Add the following if there are trigger action procedures in the routines that become invalid:

+ *number of trigger action procedures* × 5

(d) For HiRDB/Parallel Server (back-end server)

number of routines in which objects become invalid

(32) DROP [PUBLIC] VIEW

(a) For HiRDB/Single Server

10 + *number of view tables to be deleted* + *number of procedures that become invalid* + Σ (15 + *number of columns* $\times$ 4 + *number of tables that are the basis of the view table* $\times$ 4)

Add the following if there are procedures that become invalid:

+ *number of procedures that become invalid* + Σ (2 + *number of resources in procedures* $\times$ 5)

Add the following if there are trigger action procedures that reference the view tables to be deleted:

+ *number of trigger action procedures that reference the view tables to be deleted* $\times$ 5

Add the following if the table is a public view table:

+ 1
+ *number of tables having the same name as the public view table*
+ *number of view tables having the same name as the public view table*

(b) For HiRDB/Parallel Server (front-end server)

number of view tables to be deleted + *number of procedures that become invalid*

(c) For HiRDB/Parallel Server (dictionary server)

10 + Σ (15 + *number of columns* $\times$ 4 + *number of tables that are the basis of the view table* $\times$ 4)

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* + Σ (2 + *number of resources in routines* $\times$ 5)

Add the following if there are trigger action procedures that reference the view tables to be deleted:

+ *number of trigger action procedures that reference the view tables to be deleted* $\times$ 5

Add the following if the table is a public view table:

+ 1
+ *number of tables having the same name as the public view table*
+ *number of view tables having the same name as the public view table*

(d) For HiRDB/Parallel Server (back-end server)

number of routines in which objects become invalid

(33) GRANT AUDIT

(a) For HiRDB/Single Server

3

Add the following if using the password character restriction facility:

+ 2

(b) For HiRDB/Parallel Server (dictionary server)

3

Add the following if using the password character restriction facility:

+ 2

(34) GRANT CONNECT

(a) For HiRDB/Single Server

3 + *number of specified users*

Add the following if using the password character restriction facility:

+ 2

(b) For HiRDB/Parallel Server (dictionary server)

3 + *number of specified users*

Add the following if using the password character restriction facility:

+ 2

(35) GRANT DBA

(a) For HiRDB/Single Server

3 + *number of specified users*

Add the following if using the password character restriction facility:

+ 2

(b) For HiRDB/Parallel Server (dictionary server)

3 + *number of specified users*

Add the following if using the password character restriction facility:

+ 2

(36) GRANT RDAREA

(a) For HiRDB/Single Server

5 + *number of specified users*

Add the following if PUBLIC is specified:

+ *number of specified RDAREAs*

Add the following if there is a user specification:

+ *number of specified RDAREAs* × *number of specified users*

(b) For HiRDB/Parallel Server (dictionary server)

5 + *number of specified users*

Add the following if PUBLIC is specified:

+ *number of specified RDAREAs*

Add the following if there is a user specification:

+ *number of specified RDAREAs* × *number of specified users*

(37) GRANT SCHEMA

(a) For HiRDB/Single Server

3 + *number of specified users*

(b) For HiRDB/Parallel Server (dictionary server)

3 + *number of specified users*

(38) GRANT access privileges

(a) For HiRDB/Single Server

4 + *number of specified users*

(b) For HiRDB/Parallel Server (dictionary server)

4 + *number of specified users*

(39) REVOKE CONNECT

(a) For HiRDB/Single Server

3 + *number of specified users*

(b) For HiRDB/Parallel Server (dictionary server)

3 + *number of specified users*

(40) REVOKE DBA

(a) For HiRDB/Single Server

3 + *number of specified users*

(b) For HiRDB/Parallel Server (dictionary server)

3 + *number of specified users*

(41) REVOKE RDAREA

(a) For HiRDB/Single Server

10

Add the following if PUBLIC is specified:

+ *number of specified RDAREAs*

Add the following if there is a user specification:

+ *number of specified RDAREAs* × *number of specified users*

(b) For HiRDB/Parallel Server (dictionary server)

10

Add the following if PUBLIC is specified:

+ *number of specified RDAREAs*

Add the following if there is a user specification:

+ *number of specified RDAREAs* × *number of specified users*

(42) REVOKE SCHEMA

(a) For HiRDB/Single Server

5 + *number of specified users*

(b) For HiRDB/Parallel Server (dictionary server)

5 + *number of specified users*

(43) REVOKE access privileges

(a) For HiRDB/Single Server

7 + *number of specified users*

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 2 + 1

Add the following if there are view tables defined based on tables:

+ *number of single server locked resources of DROP [PUBLIC]VIEW*

(b) For HiRDB/Parallel Server (front-end server)

1

Add the following if there are routines that become invalid:

+ *number of routines that become invalid*

Add the following if there are view tables defined based on tables:

+ *number of front-end server locked resources of DROP [PUBLIC]VIEW*

(c) For HiRDB/Parallel Server (dictionary server)

6 + *number of specified users*

Add the following if there are routines that become invalid:

+ *number of routines that become invalid* × 2 + 1

Add the following if there are view tables defined based on tables:

+ *number of dictionary server locked resources of DROP [PUBLIC]VIEW*

(d) For HiRDB/Parallel Server (back-end server)

0

Add the following if there are routines that become invalid:

+ *number of routines that become invalid*

Add the following if there are view tables defined based on tables:

+ *number of back-end server locked resources of DROP [PUBLIC]VIEW*

D.2 Data manipulation SQLs

(1) SELECT (non-temporary table with LOCK TABLE not specified, WITHOUT LOCK not specified)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs storing tables to be retrieved* + *number of tables* + *number of hit retrieval rows*#3 + *number of RDAREAs storing the indexes used*#2 + *number of index keys used in retrieval*#5

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*#4

Note the following if table data is retrieved via a list:

- Determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.
- Determine the n*umber of RDAREAs storing tables to be retrieved* and the *number* of tables in the formula for the list's base tables.

- The following value must be added:
  + 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end and dictionary servers)

[preprocessing]#1

If table data is retrieved via a list, determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (back-end server)

number of RDAREAs storing tables to be retrieved + *number of tables* + *number of hit retrieval rows*#3 + *number of RDAREAs storing the indexes used*#2 + *number of index keys used in retrieval*#5

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*#4

Note the following if table data is retrieved via a list:

- Determine the n*u*mber of RDAREAs storing the tables to be retrieved and the number of tables in the formula for the list's base tables.
- The following value must be added:
  + 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: If indexes are defined for columns specified with `AND` or `OR` in a retrieval condition expression, those indexes will be used as a rule.

#3:

- If data is to be retrieved from multiple tables, determine the number of hit retrieval rows in individual tables and add that number.
- If two or more conditions are specified for index definition columns (one or more if `FOR UPDATE` is specified), determine the combined total of the rows that are retrieved based only on the conditions for each index.
- Determine the number of hit retrieval rows after excluding conditions that use the columns of multiple tables and conditions that use functions.
- Use the number of hit retrieval pages if a lock is specified for each page.

#4: For the number of logical files used by plug-ins, see the manuals included with the plug-ins.

#5: The value is `1` if index key value lock is specified, and `0` if index key value lock is not specified.

(2) SELECT (non-temporary table with LOCK TABLE not specified, WITHOUT LOCK WAIT specified)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be retrieved* + *number of tables* + 1 + *number of RDAREAs storing the indexes used*#2 + *number of index keys used in retrieval*#4

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*[#3]

Note the following if table data is retrieved via a list:

- Determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.
- Determine the n*u*mber of RDAREAs storing the tables to be retrieved and the number of tables in the formula for the list's base tables.
- The following value must be added:
  + 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end and dictionary servers)

[preprocessing][#1]

If table data is retrieved via a list, determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be retrieved + *number of tables* + 2 + *number of RDAREAs storing the indexes used*[#2] + *number of index keys used in retrieval*[#4]

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*[#3]

Note the following if table data is retrieved via a list:

- Determine the number of RDAREAs storing tables to be retrieved and the number of tables in the formula for the list's base tables.
- The following value must be added:
  + 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: If indexes are defined for columns specified with `AND` or `OR` in a retrieval condition expression, those indexes will be used as a rule.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#4: The value is `1` if index key value lock is specified, and `0` if index key value lock is not specified.

(3) SELECT (non-temporary table with LOCK TABLE not specified, WITHOUT LOCK NOWAIT specified)

(a) For HiRDB/Single Server

[preprocessing][#1] + *number of RDAREAs to be retrieved* + *number of tables* + *number of RDAREAs storing the indexes used*[#2]

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*#3

Note the following if table data is retrieved via a list:

- Determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.
- Determine the n*um*ber of RDAREAs storing tables to be retrieved and the number of tables in the formula for the list's base tables.
- The following value must be added:
  + 1

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for non-FIX tables):

+ 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end and dictionary servers)

[preprocessing]#1

If table data is retrieved via a list, determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be retrieved + *number of tables* + *number of RDAREAs storing the indexes used*#2

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*#3

Note the following if table data is retrieved via a list:

- Determine the number of RDAREAs storing tables to be retrieved and the number of tables in the formula for the list's base tables.
- The following value must be added:
  + 1

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for non-FIX tables):

+ 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: If indexes are defined for columns specified with `AND` or `OR` in a retrieval condition expression, those indexes will be used as a rule.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

(4) SELECT (temporary table, or non-temporary table with LOCK TABLE specified#1)

(a) For HiRDB/Single Server

[preprocessing]#2 + *number of RDAREAs to be retrieved* + *number of RDAREAs storing the indexes used*

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*#3

Note the following if table data is retrieved via a list:

- Determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.
- Determine the number of RDAREAs storing tables to be retrieved and the number of tables in the formula for the list's base tables.
- The following value must be added:
  + 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end and dictionary servers)

[preprocessing]#2

If table data is retrieved via a list, determine the number of lock requests that occur in the preprocessing part of the formula by applying the preprocessing formula to the list's base tables.

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be retrieved + *number of RDAREAs storing the indexes used*

Add the following if an index-type plug-in is used for retrieval or if a data-type plug-in is defined in the columns to be referenced:

+ *number of logical files used by the plug-in*#3

Note the following if table data is retrieved via a list:

- Determine the number of RDAREAs storing tables to be retrieved and the number of tables in the formula for the list's base tables.
- The following value must be added:
  + 1

Add the following if routines are used for retrieval:

+ *number of routines used* + 1

#1: If `LOCK TABLE IN SHARE MODE` is specified for `SELECT` with `FOR UPDATE` specified, the formula is the same as when `LOCK TABLE` in not specified.

#2: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

(5) INSERT (non-temporary table with INSERT to VALUES clauses specified, and LOCK TABLE not specified)

(a) For HiRDB/Single Server

[preprocessing]#1 + 5 + *number of RDAREAs to be inserted*#3 + *number of inserted VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of branched rows* + *number of indexes*

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#2

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for insertion:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

5 + *number of RDAREAs to be inserted*#3 + *number of inserted VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of branched rows* + *number of indexes*

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#2

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#3: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

(6) INSERT (non-temporary table with INSERT to VALUES clauses specified, and LOCK TABLE specified#1)

(a) For HiRDB/Single Server

[preprocessing]#2 + 1 + *number of RDAREAs to be inserted*#4

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 5

+ 2 × *number of temporary table indexes*

+ 3 × *number of temporary table RDAREAs within the single server*

+ *total number of segments used by temporary tables and temporary table indexes*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#2

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#2

(d) For HiRDB/Parallel Server (back-end server)

1 + *number of RDAREAs to be inserted*#4

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 5
+ 2 × *number of temporary table indexes*
+ 3 × *number of temporary table RDAREAs within the back-end server*
+ *total number of segments used by temporary tables and temporary table indexes*

#1: If `LOCK TABLE IN SHARE MODE` is specified for a non-temporary table, the formula is the same as when `LOCK TABLE` is not specified.

#2: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#4: Total number of RDAREAs for the target indexes, LOB columns, and LOB attributes.

(7) INSERT (INSERT to SELECT clauses specified)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be inserted*#5 + 4 + *number of inserted rows*#2, #6 + *number of columns of inserted rows of VARCHAR, NVARCHAR, or MVARCHAR that are 256 bytes or larger* × *number of inserted rows*#2, #6 × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of inserted rows*#2, #6 × *number of branched rows* + *number of inserted rows*#6 × *number of indexes* + 1 + *number of RDAREAs to be retrieved*#5 + *number of tables to be retrieved* + *number of hit retrieval rows*#3, #6

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-ins to be inserted*#4 + *number of logical files used by the plug-ins used for retrieval*#4

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 2 + 2 × *number of temporary table indexes*

+ *number of temporary table RDAREAs within the single server* × 3

+ *total number of segments used by temporary tables and temporary table indexes*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

Add the following if routines are used for insertion:

+ *number of routines used* + 1

(d) For HiRDB/Parallel Server (back-end server)

4 + *number of RDAREAs to be inserted*#5 + *number of inserted rows*#2, #6 + *number of columns of inserted rows of VARCHAR, NVARCHAR, or MVARCHAR that are 256 bytes or larger* × *number of inserted rows*#6 × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of inserted rows*#6 × *number of branched rows* + *number of inserted rows*#6 × *number of indexes* + 1 + *number of RDAREAs to be retrieved*#5 + *number of tables to be retrieved* + *number of hit retrieval rows*#3, #6

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-ins to be inserted*#4 + *number of logical files used by the plug-ins used for retrieval*#4

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 2 + 2 × *number of temporary table indexes*

+ *number of temporary table RDAREAs within the back-end server* × 3

+ *total number of segments used by temporary tables and temporary table indexes*

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Use the number of pages inserted if locking by page is specified.

#3:

- If data is to be retrieved from multiple tables, determine the number of hit retrieval rows in individual tables and add that number.
- If two or more conditions are specified for index definition columns, determine the combined total of the rows that are retrieved based only on the conditions for each index.

- Determine the number of hit retrieval rows after excluding conditions that use the columns of multiple tables and conditions that use functions.
- Use the number of hit retrieval pages if a lock is specified for each page.

#4: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#5: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

#6: For a temporary table, the value is 0.

(8) INSERT (non-temporary table with INSERT to VALUES clauses specified, LOCK TABLE not specified, and index key value no locking used)

(a) For HiRDB/Single Server

[preprocessing]#1 + 5 + *number of RDAREAs to be inserted*#2 + *number of inserted VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of branched rows* + 1

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for insertion:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

5 + *number of RDAREAs to be inserted*#2 + *number of inserted VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of branched rows* + 1

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

#3: For the number of logical files used by each plug-in, refer to the manual provided with the plug-in.

(9) INSERT (temporary table, or INSERT to VALUES clauses specified, LOCK TABLE specified#4, and index key value no locking used)

(a) For HiRDB/Single Server

[preprocessing]#1 + 1 + *number of RDAREAs to be inserted*#2

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 5
+ 2 × *number of temporary table indexes*
+ 3 × *number of temporary table RDAREAs within the single server*
+ *total number of segments used by temporary tables and temporary table indexes*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

1 + *number of RDAREAs to be inserted*#2

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 5
+ 2 × *number of temporary table indexes*
+ 3 × *number of temporary table RDAREAs within the back-end server*
+ *total number of segments used by temporary tables and temporary table indexes*

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Total number of RDAREAs for the target indexes, LOB columns, and LOB attributes.

#3: For the number of logical files used by each plug-in, refer to the manual provided with the plug-in.

#4: If `LOCK TABLE IN SHARE MODE` is specified for a non-temporary table, the formula is the same as when `LOCK TABLE` is not specified.

(10) INSERT (INSERT to SELECT clauses specified, index key value no locking used).

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be inserted*#2 + 4 + *number of inserted rows*#3, #6 + *number of inserted VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of inserted rows*#6 × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of inserted rows*#6 × *number of branched rows* + *number of indexes* + 1 + *number of RDAREAs to be retrieved*#2 + *number of tables to be retrieved* + *number of hit retrieval rows*#4, #6

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-ins to be inserted*#5 + *number of logical files used by the plug-ins used for retrieval*#5

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 2

+ 2 × *number of temporary table indexes*

+ 3 × *number of temporary table RDAREAs within the server*

+ *total number of segments used by temporary tables and temporary table indexes*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if inserting to an SQL session-specific temporary table:

+ *number of SQL session-specific temporary tables that are used*

Add the following if inserting to a temporary table:

+ 2

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

Add the following if routines are used for insertion:

+ *number of routines used* + 1

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be inserted#2 + 4 + *number of inserted rows*#3, #6 + *number of inserted VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of inserted rows*#6 × *number of branched rows* + *number of inserted BINARY-type columns that cannot be stored in one page* × *number of inserted rows*#6 × *number of branched rows* + *number of indexes* + 1 + *number of RDAREAs to be retrieved*#2 + *number of tables to be retrieved* + *number of hit retrieval rows*#4, #6

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-ins to be inserted*#5 + *number of logical files used by the plug-ins used for retrieval*#5

Add the following if routines are used for insertion:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for insertion:

+ *number of sequence generators used* × 2

Add the following if inserting to a temporary table:

+ 2

+ 2 × *number of temporary table indexes*

+ 3 × *number of temporary table RDAREAs within the back-end server*

+ *total number of segments used by temporary tables and temporary table indexes*

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

#3: Use the number of inserted pages if a lock is specified for each page.

#4:

- If data is to be retrieved from multiple tables, determine the number of hit retrieval rows in individual tables and add that number.
- If two or more conditions are specified for index definition columns, determine the combined total of the rows that are retrieved based only on the conditions for each index.
- Determine the number of hit retrieval rows after excluding conditions that use the columns of multiple tables and conditions that use functions.
- Use the number of hit retrieval pages if a lock is specified for each page.

#5: For the number of logical files used by each plug-in, refer to the manual provided with the plug-in.

#6: For a temporary table, the value is 0.

(11) UPDATE (non-temporary table with LOCK TABLE not specified)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be updated*#5 + 4 + *number of updated rows*#2 + *number of pre-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of post-update rows* × *number of branched rows after update* + *number of pre-update BINARY-type columns that cannot be stored in one page* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update BINARY-type columns that cannot be stored in one page* × *number of post-update rows* × *number of branched rows after update* + *number of indexes to be updated* × 2 × *number of updated rows* + Σ (*number of segments storing pre-update LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*#3

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* × 2

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for updating:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be updated#5 + 4 + *number of updated rows*#2 + *number of pre-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of post-update rows* × *number of branched rows after update* + *number of pre-update BINARY-type columns that cannot be stored in one page* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update BINARY-type columns that cannot be stored in one page* × *number of post-update rows* × *number of branched rows after update* + *number of indexes to be updated* × 2 × *number of updated rows* + Σ (*number of segments storing pre-update LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*#3

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* × 2

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2

- If there is a conditional expression that needs to retrieve data from another table, determine the number of hit retrieval rows for that table and add it to the number of updated rows.
- If conditions are specified for an index definition column, determine the combined total of the rows that are retrieved based only on the conditions for each index. Then, use the number of hit retrieval rows as the number of updated rows.
- Determine the number of hit retrieval rows after excluding conditions that use the columns of multiple tables and conditions that use functions. Then, use the number of hit retrieval rows as the number of updated rows.
- Use the number of updated pages if a lock is specified for each page.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#4: Repeat and add this for the number of LOB columns and LOB attributes.

#5: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

(12) UPDATE (temporary table or LOCK TABLE is specified[#1])

(a) For HiRDB/Single Server

[preprocessing][#2] + *number of RDAREAs to be updated*[#5] + 1 + Σ (*number of segments storing pre-update LOB data*)[#4]

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*[#3]

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* $\times$ 2

Add the following if temporary tables are used for updating:

+ *total number of segments used by temporary tables and temporary table indexes*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing][#2]

Add the following if routines are used for updating:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing][#2]

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be updated[#5] + 1 + Σ (*number of segments storing pre-update LOB data*)[#4]

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*[#3]

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* $\times$ 2

Add the following if temporary tables are used for updating:

+ *total number of segments used by temporary tables and temporary table indexes*

#1: If `LOCK TABLE IN SHARE MODE` is specified for a non-temporary table, the formula is the same as when `LOCK TABLE` is not specified.

#2: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#4: Repeat and add this for the number of LOB columns and LOB attributes.

#5: Total number of RDAREAs for the target indexes, LOB columns, and LOB attributes.

(13) UPDATE (non-temporary table with LOCK TABLE not specified and index key value no locking used)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be updated*#2 + 4 + *number of updated rows*#3 + *number of pre-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of post-update rows* × *number of branched rows after update* + *number of pre-update BINARY-type columns that cannot be stored in one page* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update BINARY-type columns that cannot be stored in one page* × *number of post-update rows* × *number of branched rows after update* + Σ (*number of segments storing pre-update LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*#5

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* × 2

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for updating:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be updated#2 + 4 + *number of updated rows*#3 + *number of pre-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update VARCHAR, NVARCHAR, or MVARCHAR columns that are 256 bytes or larger* × *number of post-update rows* × *number of branched rows after update* + *number of pre-update BINARY-type columns that cannot be stored in one page* × *number of pre-update rows* × *number of branched rows before update* + *number of post-update BINARY-type columns that cannot be stored in one page* × *number of post-update rows* × *number of branched rows after update* + Σ (*number of segments storing pre-update LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*#5

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* × 2

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

#3

- If there is a conditional expression that needs to retrieve data from another table, determine the number of hit retrieval rows for that table and add it to the number of updated rows.
- If conditions are specified for an index definition column, determine the combined total of the rows that are retrieved based only on the conditions for each index. Then, use the number of hit retrieval rows as the number of updated rows.
- Determine the number of hit retrieval rows after excluding conditions that use the columns of multiple tables and conditions that use functions. Then, use the number of hit retrieval rows as the number of updated rows.
- Use the number of updated pages if a lock is specified for each page. If a unique index is defined for the table, also add the value for the rows to be updated.

#4: Repeat and add this for the number of LOB columns and LOB attributes.

#5: For the number of logical files used by each plug-in, refer to the manual provided with the plug-in.

(14) UPDATE (temporary table, or LOCK TABLE specified#5 and index key value no locking used)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be updated*#2 + 1 + Σ (*number of segments storing pre-update LOB data*)#3

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*#4

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* × 2

Add the following if temporary tables are used for updating:

+ *total number of segments used by temporary tables and temporary table indexes*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for updating:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be updated#2 + 1 + Σ (*number of segments storing pre-update LOB data*)#3

Add the following if a data-type plug-in or index-type plug-in is defined for the columns to be updated:

+ *number of logical files used by the plug-in*#4

Add the following if LOCK is specified for the pd_nowait_scan_option operand (add only for a non-FIX table):

+ 1

Add the following if routines are used for updating:

+ *number of routines used* + 1

Add the following if sequence generators (NEXT VALUE expressions) are used for updating:

+ *number of sequence generators used* × 2

Add the following if temporary tables are used for updating:

+ *total number of segments used by temporary tables and temporary table indexes*

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Total number of RDAREAs for the indexes, LOB columns, and LOB attributes.

#3: Repeat and add this for the number of LOB columns and LOB attributes.

#4: For the number of logical files used by each plug-in, refer to the manual provided with the plug-in.

#5: If `LOCK TABLE IN SHARE MODE` is specified for a non-temporary table, the formula is the same as when `LOCK TABLE` is not specified.

(15) DELETE (non-temporary table with LOCK TABLE not specified)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be deleted*#5 + 1 + *number of deleted rows*#2 + *number of VARCHAR, NVARCHAR, or MVARCHAR columns deleted that are 256 bytes or larger* × *number of branched rows* × *number of deleted rows* + *number of BINARY-type columns to be deleted that cannot be stored in one page* × *number of branched rows* × *number of deleted rows* + *number of indexes* × *number of deleted rows* + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be deleted#5 + 1 + *number of deleted rows*#2 + *number of VARCHAR, NVARCHAR, or MVARCHAR columns deleted that are 256 bytes or larger* × *number of branched rows* × *number of deleted rows* + *number of BINARY-type columns to be deleted that cannot be stored in one page* × *number of branched rows* × *number of deleted rows* + *number of indexes* × *number of deleted rows* + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for deletion:

+ *number of routines used* + 1

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2

- If there is a conditional expression that needs to retrieve data from another table, determine the number of hit retrieval rows for that table and add that value to the number of deleted rows.

- If conditions are specified for index definition columns, determine the combined total of the rows that are retrieved based only on the conditions for each index. Then, use the number of hit retrieval rows as the number of deleted rows.
- Determine the number of hit retrieval rows after excluding conditions that use the columns of multiple tables and conditions that use functions. Then, use the number of hit retrieval rows as the number of deleted rows.
- Use the number of deleted pages if a lock is specified for each page.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#4: Repeat and add this for the number of LOB columns and LOB attributes.

#5: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

(16) DELETE (temporary table, or LOCK TABLE specified and index key value no locking used#1)

(a) For HiRDB/Single Server

[preprocessing]#2 + *number of RDAREAs to be deleted*#5 + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)#4 + *number of released data pages*#6

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#2

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#2

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be deleted#5 + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)#4 + *number of released data pages*#6

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#3

Add the following if routines are used for deletion:

+ *number of routines used* + 1

#1: If `LOCK TABLE IN SHARE MODE` is specified for a non-temporary table, the formula is the same as when `LOCK TABLE` is not specified.

#2: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#3: For the number of logical files used by plug-ins, refer to the manuals included with the plug-ins.

#4: Repeat and add this for the number of LOB columns and LOB attributes.

#5: Total number of RDAREAs for the target indexes, LOB columns, and LOB attributes.

#6: Number of pages from which all data is deleted.

(17) DELETE (non-temporary table with LOCK TABLE not specified and index key value no locking used)

(a) For HiRDB/Single Server

[preprocessing]#1 + *number of RDAREAs to be deleted*#2 + 1 + *number of deleted rows*#3 + *number of VARCHAR, NVARCHAR, or MVARCHAR columns deleted that are 256 bytes or larger* × *number of branched rows* × *number of deleted rows* + *number of BINARY-type columns to be deleted that cannot be stored in one page* × *number of branched rows* × *number of deleted rows* + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#5

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be deleted#2 + 1 + *number of deleted rows*#3 + *number of VARCHAR, NVARCHAR, or MVARCHAR columns deleted that are 256 bytes or larger* × *number of branched rows* × *number of deleted rows* + *number of BINARY-type columns that cannot be stored in one page* × *number of branched rows* + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)#4

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*#5

Add the following if routines are used for deletion:

+ *number of routines used* + 1

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

#3

- If there is a conditional expression that needs to retrieve data from another table, determine the number of hit retrieval rows for that table and add it to the number of deleted rows.
- If conditions are specified for an index definition column, determine the combined total of the rows that are retrieved based only on the conditions for each index. Then, use the number of hit retrieval rows as the number of deleted rows.
- Determine the number of hit retrieval rows after excluding conditions that use the columns of multiple tables and conditions that use functions. Then, use the number of hit retrieval rows as the number of deleted rows.
- Use the number of deleted pages if a lock is specified for each page. If a unique index is defined for the table, also add the value for the rows to be deleted.

#4: Repeat and add this for the number of LOB columns and LOB attributes.

#5: For the number of logical files used by each plug-in, refer to the manual provided with the plug-in.

(18) DELETE (temporary table, or LOCK TABLE specified[#6] and index key value no locking used)

(a) For HiRDB/Single Server

[preprocessing][#1] + *number of RDAREAs to be deleted*[#2] + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)[#3] + *number of released data pages*[#4]

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*[#5]

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing][#1]

Add the following if routines are used for deletion:

+ *number of routines used* + 1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing][#1]

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs to be deleted[#2] + Σ (*number of deleted LOB data segments* + *number of segments storing the deleted LOB data*)[#3] + *number of released data pages*[#4]

Add the following if a data-type plug-in or index-type plug-in is defined:

+ *number of logical files used by the plug-in*[#5]

Add the following if routines are used for deletion:

+ *number of routines used* + 1

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Total number of RDAREAs for the target indexes, LOB columns, and LOB attributes.

#3: Repeat and add this for the number of LOB columns and LOB attributes.

#4: Number of pages from which all data is deleted.

#5: For the number of logical files used by each plug-in, refer to the manual provided with the plug-in.

#6: If `LOCK TABLE IN SHARE MODE` is specified for a non-temporary table, the formula is the same as when `LOCK TABLE` is not specified.

(19) PURGE TABLE

(a) For HiRDB/Single Server

[preprocessing][#1] + *number of target RDAREAs*[#3] + 3 + *number of table segments being used* × 2 + Σ ((*number of index segments being used* × 2) + 2)[#2] + Σ (↑ *number of segments for which LOB data is stored in HiRDB files* ÷ 64,000 ↑)[#4]

Add the following if USE is specified in the pd_check_pending operand or the operand is omitted:

+ 4

Also add the following if the target table is a referenced table:

+ 3 + *number of table-storage RDAREAs for the target table* + *number of referencing tables that reference the target table* × 2 + Σ (1 + *number of table-storage RDAREAs for referencing tables that reference the target table*)[#5] + Σ (*number of target RDAREAs of the referencing tables*[#3] + 2)[#5]

- **Also add the following if the target table is a partitioned table:**
  + 1 + *number of table-storage RDAREAs for the target table*
- **Also add the following if a referencing table that references the target table is a partitioned table:**
  + Σ (1 + *number of table-storage RDAREAs for referencing tables that reference the target table*)#5

Also add the following if the target table is a referencing table or a table that defines a check constraint:

+ 1 + *number of referential constraints defined in the target table* + 1 + *number of check constraints defined in the target table*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]#1

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]#1

Add the following if USE is specified in the pd_check_pending operand or the operand is omitted:

+ 4

Also add the following if the target table is a referenced table:

+ 3 + *number of table-storage RDAREAs for the target table* + *number of referencing tables that reference the target table* × 2 + Σ (1 + *number of table-storage RDAREAs for referencing tables that reference the target table*)#5

- **Also add the following if the target table is a partitioned table:**
  + 1 + *number of table-storage RDAREAs for the target table*
- **Also add the following if a referencing table that references the target table is a partitioned table:**
  + Σ (1 + *number of table-storage RDAREAs for referencing tables that reference the target table*)#5

Also add the following if the target table is a referencing table or a table that defines a check constraint:

+ 1 + *number of referential constraints defined in the target table* + 1 + *number of check constraints defined in the target table*

(d) For HiRDB/Parallel Server (back-end server)

number of target RDAREAs#3 + 3 + *number of table segments being used* × 2 + Σ ((*number of index segments being used* × 2) + 2)#2 + Σ (↑ *number of segments for which LOB data is stored in HiRDB files* ÷ 64,000 ↑)#4

Add the following if USE is specified in the pd_check_pending operand or the operand is omitted, and if the target table is a referenced table:

+ Σ (*number of target RDAREAs of the referencing table*#3 + 2)#5

#1: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

#2: Repeat and add this for the number of indexes.

#3: Total number of RDAREAs for the target tables, indexes, LOB columns, and LOB attributes.

#4: Repeat and add this for the number of HiRDB files for LOB RDAREAs of LOB columns and LOB attributes.

#5: Iteratively add the number of referencing tables that reference the target table.

(20) ASSIGN LIST statement (list creation from a base table)

(a) For HiRDB/Single Server

number of base table SELECT lock requests#1 + 7 + 2

Add the following value if an existing list is specified as the name of the list to be created:

+ *number of RDAREAs for the base table of the list before re-creation* + 3 + *number of data segments of the list before re-creation*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]$^{\#2}$

(c) For HiRDB/Parallel Server (dictionary server)

preprocessing$^{\#2}$ + 7

Add the following value if an existing list is specified as the name of the list to be created:

+ *number of RDAREAs for the base table of the list before re-creation*

(d) For HiRDB/Parallel Server (back-end server)

number of base table SELECT lock requests$^{\#1}$ + 2

Add the following value if an existing list is specified as the name of the list to be created:

+ 3 + *number of data segments of the list before re-creation*

#1: For details about the number of base table `SELECT` lock requests, see the `SELECT` statement in *(1) SELECT (non-temporary table with LOCK TABLE not specified, WITHOUT LOCK not specified)*, *(2) SELECT (non-temporary table with LOCK TABLE not specified, WITHOUT LOCK WAIT specified)*, *(3) SELECT (non-temporary table with LOCK TABLE not specified, WITHOUT LOCK NOWAIT specified)*, or *(4) SELECT (temporary table, or non-temporary table with LOCK TABLE specified)* that matches the conditions of the base table.

#2: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

(21) ASSIGN LIST statement (list creation from a list)

(a) For HiRDB/Single Server

[preprocessing]$^{\#}$ + 7 + *number of RDAREAs storing the base table of the list* + 4

If an existing name is specified for the list to be created, add the following value:

+ *number of RDAREAs storing the base table of the list before re-creation* + 3 + *number of segments using the list data before re-creation*

(b) For HiRDB/Parallel Server (front-end server)

[preprocessing]$^{\#}$

(c) For HiRDB/Parallel Server (dictionary server)

[preprocessing]$^{\#}$ + 7

If an existing name is specified for the list to be created, add the following value:

+ *number of RDAREAs storing the base table of the list before re-creation*

(d) For HiRDB/Parallel Server (back-end server)

number of RDAREAs storing the base table of the list + 4

If an existing name is specified for the list to be created, add the following value:

+ 3 + *number of segments using the list data before re-creation*

#: For the number of lock requests in preprocessing, see *(25) Number of lock requests in preprocessing*.

(22) ASSIGN LIST statement (list name change)

(a) For HiRDB/Single Server

7

If an existing name is specified for the list to be created, add the following value:

+ *number of RDAREAs storing the base table of the list before re-creation*

(b) For HiRDB/Parallel Server (dictionary server)

7

If an existing name is specified for the list to be created, add the following value:

+ *number of RDAREAs storing the base table of the list before re-creation*

(23) DROP LIST statement (deletion with list name specification: DROP LIST ~)

(a) For HiRDB/Single Server

7 + *number of RDAREAs storing the base table of the list* + 3 + *number of segments using the list data*

(b) For HiRDB/Parallel Server (dictionary server)

7 + *number of RDAREAs storing the base table of the list*

(c) For HiRDB/Parallel Server (back-end server)

3 + *number of segments using the list data*

(24) DROP LIST statement (deletion of all lists locally owned: DROP ALL LIST ~)

(a) For HiRDB/Single Server

1 + *number of RDAREAs for lists* + *number of deleted lists* × 2 + *total number of segments using the deleted list data*

(b) For HiRDB/Parallel Server (dictionary server)

1

(c) For HiRDB/Parallel Server (back-end server)

Number of RDAREAs for lists + *number of deleted lists* × 2 + *total number of segments using the deleted list data*

(25) Number of lock requests in preprocessing

(a) For HiRDB/Single Server

1 + *number of tables used* + *number of view tables used* + Σ#1 (2 + *number of columns* × 2 + *number of indexes* × 2 + *number of table partitions* × 4 + Σ#2 (*number of partitioning indexes* × 2) + 2) + 13

Add the following if a user-defined type or function is used:

+ 1

Add the following (for each table) if optimization information is collected:

+ 1 + *number of columns for which optimization information is created* × 4 + 1

Add the following (for each view table) if view tables are used:

+ Σ (*number of columns* × 2 + 4 + *number of columns comprising the view table* + 2)

Add the following if LOB **columns are defined:**

+ *number of LOB columns* × 2 + *number of table partitions*

Add the following if user-defined type columns are defined:

number of user-defined type columns + Σ (4 + *number of attributes* × 2)

Add the following if LOB attributes are used:

+ *number of LOB attributes* × 2 + *number of table partitions*

Add the following if a higher-order type is used:

+ 2

Add the following for each function used if functions are used:

+ Σ (2 + *number of functions with the same name and parameter configuration* × 2 + *number of parameters* × 2)

Add the following for each plug-in used if plug-ins are used:

+ Σ (2 + *number of parameters* × 2) + 3 + *number of contexts* × 4 + 2

Add the following if routines are used:

+ *number of routines used* + 1

Add the following if sequence generators are used:

+ *number of sequence generators used* + 1 + *number of sequence generators used*

(b) For HiRDB/Parallel Server (front-end server)

1 + *number of tables used* + *number of view tables used*

Add the following if a user-defined type or function is used:

+ 1

Add the following if routines are used:

+ *number of routines used* + 1

Add the following if sequence generators are used:

+ *number of sequence generators used*

(c) For HiRDB/Parallel Server (dictionary server)

$\Sigma^{\#1}$ (2 + *number of columns* × 2 + *number of indexes* × 2 + *number of table partitions* × 4 + $\Sigma^{\#2}$ (*number of partitioning indexes* × 2) + 2) + 13

Add the following (for each table) if optimization information is collected:

+ 1 + *number of columns for which optimization information is created* × 4 + 1

Add the following (for each view table) if view tables are used:

+ Σ (*number of columns* × 2 + 4 + *number of columns comprising the view table* + 2)

Add the following if LOB columns are defined:

+ *number of LOB columns* × 2 + *number of table partitions*

Add the following for each user-defined type column if user-defined type columns are defined:

+ Σ (4 + *number of attributes* × 2)

Add the following if LOB attributes are used:

+ *number of LOB attributes* × 2 + *number of table partitions*

Add the following if a higher-order type is used:

+ 2

Add the following for each function used if functions are used:

+ Σ (2 + *number of functions with the same name and parameter configuration* × 2 + *number of parameters* × 2)

Add the following for each plug-in used if plug-ins are used:

+ Σ (2 + *number of parameters* × 2) + 3 + *number of contexts* × 4 + 2

Add the following if sequence generators are used:

+ 1 + *number of sequence generators used*

#1: Summarize this for each table.

#2: Summarize this for each partitioning index inside the table.

D.3 Control SQL

(1) LOCK

(a) For HiRDB/Single Server

[preprocessing]# + *number of table row partitions* + 1

(b) For HiRDB/Parallel Server (front-end server and dictionary server)

[preprocessing]#

(c) For HiRDB/Parallel Server (back-end server)

Number of table row partitions + 1

#: For the number of lock requests in preprocessing, see in Section *D.2(25) Number of lock requests in preprocessing*.

D.4 Utilities and commands

(1) Database load utility (pdload)

(a) For HiRDB/Single Server

209 + α + *number of segments used by RDAREAs for table* + *number of segments used by RDAREAs for index* + (↑ *number of segments used by LOB RDAREAs* ÷ 64,000 ↑)

(b) For HiRDB/Parallel Server (front-end server and dictionary server)

3

(c) For HiRDB/Parallel Server (dictionary server)

206 + α

(d) For HiRDB/Parallel Server (back-end server)

Number of segments used by RDAREAs for table + *number of segments used by RDAREAs for index* + (↑ *number of segments used by LOB RDAREAs* ÷ 64,000 ↑)

α : MAX(*number of table columns*, *number of table indexes*, *number of RDAREAs for table*)

Determine the number of segments used from the number of segments scheduled to store data. When the `-d` option is specified and if the number of segments already storing data (the number of segments being used) is greater than the number of segments scheduled to store data, use for computation the number of segments being used.

(2) Database reorganization utility (pdrorg)

(a) For HiRDB/Single Server

209 + α + *number of segments used by RDAREAs* for table + *number of segments used by RDAREAs for index* (↑ *number of segments used by LOB RDAREAs* ÷ 64,000 ↑)

(b) For HiRDB/Parallel Server (front-end server and dictionary server)

3

(c) For HiRDB/Parallel Server (dictionary server)

206 + α

(d) For HiRDB/Parallel Server (back-end server)

Number of segments used by RDAREAs for table + *number of segments used by RDAREAs for index* + ($\uparrow$ *number of segments used by LOB RDAREAs* $\div$ 64,000 $\uparrow$)

α : MAX(*number of table columns*, *number of table indexes*, *number of RDAREAs for table*)

(3) Rebalancing utility (pdrbal)

(a) For HiRDB/Single Server

Shared mode (`-k share`)

starting/finishing$^{\#1}$ + preprocessing$^{\#2}$ + rebalancing$^{\#3}$ + 2

Exclusive mode (`-k exclusive`)

starting/finishing$^{\#1}$ + preprocessing^{2} + rebalancing$^{\#3}$

(b) For HiRDB/Parallel Server

Shared mode (`-k share`)

- Front-end server: starting/finishing$^{\#1}$ + preprocessing$^{\#2}$ + 1
- Dictionary server: starting/finishing$^{\#1}$ + preprocessing$^{\#2}$
- Back-end server: starting/finishing$^{\#1}$ + rebalancing$^{\#3}$

Exclusive mode (`-k exclusive`)

- Front-end server: starting/finishing$^{\#1}$ + preprocessing$^{\#2}$
- Dictionary server: starting/finishing$^{\#1}$ + preprocessing$^{\#2}$
- Back-end server: starting/finishing$^{\#1}$ + rebalancing$^{\#3}$

#1: Number of lock requests for starting/finishing = $\Sigma A_i + \Sigma B_i$

A_i: Number of lock requests per function

For the number of lock requests of a function, see in *D.1(10) CREATE [PUBLIC] FUNCTION*.

B_i: Number of lock requests per procedure

For the number of lock requests of a procedure, see in *D.1(13) CREATE [PUBLIC] PROCEDURE*.

#2: The number of lock requests for preprocessing is as follows:

- HiRDB/Single Server: 225
- HiRDB/Parallel Server (front-end server): 5
- HiRDB/Parallel Server (dictionary server): 220

#3: The number of lock requests for rebalancing = *number of segments used by RDAREAs for table* + *number of segments used by RDAREAs for index* + ($\uparrow$ *number of segments used by LOB RDAREAs* $\div$ 64,000 $\uparrow$)

(4) pdacunlck command

(a) For HiRDB/Single Server

0

Add the following if an authorization identifier is specified:

+ 128

Add the following if ALL is specified:

+ *number-of-users-registered-in-dictionary-table-SQL_USERS*

(b) For HiRDB/Parallel Server (dictionary server)

0

Add the following if an authorization identifier is specified:

+ 128

Add the following if ALL is specified:

+ *number-of-users-registered-in-dictionary-table-SQL_USERS*

E. Operands Checked by the pdconfchk Command

The table below shows the operands that are checked by the `pdconfchk` command (the operands are listed in alphabetical order). Note that the operands of the following definitions are not checked by the `pdconfchk` command.

- UAP environment definition

Table E–1: Operands checked by the pdconfchk command

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_additional_optimize_level	Y	N	N	N	N	Y
pd_alv_port	Y	N	N	N	N	Y
pd_apply_search_ats_num	Y	N	N	N	N	Y
pd_assurance_index_no	Y	N	N	N	N	Y
pd_assurance_table_no	Y	N	N	N	N	Y
pd_audit	Y	N	N	N	N	Y
pd_aud_async_buff_count	Y	N	N	N	N	Y
pd_aud_async_buff_retry_intvl	Y	N	N	N	N	N
pd_aud_async_buff_size	Y	N	N	N	N	Y
pd_aud_auto_loading	Y	N	N	N	N	Y
pd_aud_file_name	Y	N	N	N	N	Y
pd_aud_file_wrn_pnt	Y	N	N	N	N	Y
pd_aud_max_generation_num	Y	N	N	N	N	Y
pd_aud_max_generation_size	Y	N	N	N	N	Y
pd_aud_no_standby_file_opr	Y	N	N	N	N	Y
pd_aud_sql_data_size	Y	N	N	N	N	N
pd_aud_sql_source_size	Y	N	N	N	N	N
pd_audit_def_buffer_size	Y	N	N	N	N	N
pd_auth_cache_size	Y	N	N	N	N	Y
pd_auto_vrup	Y	N	N	N	N	Y
pd_bes_connection_hold	Y	N	N	N	N	N
pd_bes_conn_hold_trn_interval	Y	N	N	N	N	N
pd_bes_shmpool_size	Y	N	N	N	N	Y
pd_c_library_directory	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_cancel_down_msgchange	Y	N	N	N	N	Y
pd_cancel_dump	Y	N	N	N	N	Y
pd_change_clt_ipaddr	Y	N	N	N	N	Y
pd_check_pending	Y	N	N	N	N	N
pd_client_waittime_over_abort	Y	N	N	N	N	Y
pd_clt_waittime_over_dump_level	Y	N	N	N	N	Y
pd_cmdhold_precheck	Y	N	N	N	N	Y
pd_command_deadlock_priority	Y	N	N	N	N	Y
pd_connect_errmsg_hide	Y	N	N	N	N	Y
pd_constraint_name	Y	N	N	N	N	N
pd_cwaittime_report_dir	Y	N	N	N	N	N
pd_cwaittime_report_size	Y	N	N	N	N	N
pd_cwaittime_wrn_pnt	Y	N	N	N	N	N
pd_dbbuff_attribute	Y	N	N	N	N	Y
pd_dbbuff_binary_data_lru	Y	N	N	N	N	Y
pd_dbbuff_lock_interval	Y	N	N	N	N	Y
pd_dbbuff_lock_release_detect	Y	N	N	N	N	Y
pd_dbbuff_lock_spn_count	Y	N	N	N	N	Y
pd_dbbuff_lru_option	Y	N	N	N	N	Y
pd_dbbuff_modify	Y	N	N	N	N	Y
pd_dbbuff_rate_updpage	Y	N	N	N	N	Y
pd_dbbuff_trace_level	Y	N	N	N	N	Y
pd_dbbuff_wait_interval	Y	N	N	N	N	Y
pd_dbbuff_wait_spn_count	Y	N	N	N	N	Y
pd_dbreuse_remaining_entries	Y	N	N	N	N	Y
pd_dbsync_altwrite_skip	Y	N	N	N	N	Y
pd_dbsync_lck_release_count	Y	N	N	N	N	Y
pd_dbsync_point	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
`pd_db_access_error_action`	Y	N	N	N	N	Y
`pd_db_hold_action`	Y	N	N	N	N	Y
`pd_db_io_error_action`	Y	N	N	N	N	N
`pd_deadlock_priority_use`	Y	N	N	N	N	Y
`pd_debug_info_netstat`	Y	N	N	N	N	Y
`pd_dec_sign_normalize`	Y	N	N	N	N	Y
`pd_def_buf_control_area_assign`	N	N	N	N	N	N
`pd_delete_reserved_word_file`	Y	N	N	N	N	Y
`pd_deter_restart_on_stop_fail`	Y	N	N	N	N	Y
`pd_dfw_awt_process`	Y	N	N	N	N	Y
`pd_dfw_syncpoint_skip_limit`	Y	N	N	N	N	N
`pd_dic_shmpool_size`	Y	N	N	N	N	Y
`pd_down_watch_proc`	Y	N	N	N	N	Y
`pd_dump_suppress_watch_time`	Y	N	N	N	N	Y
`pd_fes_lck_pool_partition`	Y	N	N	N	N	Y
`pd_fes_lck_pool_size`	Y	N	N	N	N	Y
`pd_floatable_bes`	Y	N	N	N	N	N
`pd_foreign_server_libpath`	Y	N	N	N	N	N
`pd_ha`	Y	N	N	N	N	Y
`pd_ha_acttype`	Y	N	N	N	N	N
`pd_ha_agent`	Y	N	N	N	N	N
`pd_ha_ipaddr_inherit`	Y	N	N	N	N	Y
`pd_ha_max_act_guest_servers`	Y	N	N	N	N	Y
`pd_ha_max_server_process`	Y	N	N	N	N	Y
`pd_ha_mgr_rerun`	Y	N	N	N	N	Y
`pd_ha_process_count`	Y	N	N	N	N	Y
`pd_ha_resource_act_wait_time`	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_ha_server_process_standby	Y	N	N	N	N	N
pd_ha_switch_timeout	Y	N	N	N	N	Y
pd_ha_transaction	Y	N	N	N	N	Y
pd_ha_trn_queuing_wait_time	Y	N	N	N	N	Y
pd_ha_trn_restart_retry_time	Y	N	N	N	N	Y
pd_ha_unit	Y	N	N	N	N	N
pd_hash_table_size	Y	N	N	N	N	Y
pd_hashjoin_hashing_mode	Y	N	N	N	N	Y
pd_hostname	Y	N	N	N	N	Y
pd_host_watch_interval	Y	N	N	N	N	Y
pd_indexlock_mode	Y	N	N	N	N	Y
pd_inet_unix_bufmode	Y	N	N	N	N	Y
pd_inet_unix_utl_bufmode	Y	N	N	N	N	Y
pd_ipc_clt_conn_nblock	Y	N	N	N	N	N
pd_ipc_clt_conn_nblock_time	Y	N	N	N	N	N
pd_ipc_conn_count	Y	N	N	N	N	Y
pd_ipc_conn_interval	Y	N	N	N	N	Y
pd_ipc_conn_nblock	Y	N	N	N	N	Y
pd_ipc_conn_nblock_time	Y	N	N	N	N	Y
pd_ipc_inet_bufsize	Y	N	N	N	N	Y
pd_ipc_recv_count	Y	N	N	N	N	Y
pd_ipc_send_count	Y	N	N	N	N	Y
pd_ipc_send_retrycount	Y	N	N	N	N	Y
pd_ipc_send_retrysleeptime	Y	N	N	N	N	Y
pd_java_archive_directory	Y	N	N	N	N	Y
pd_java_castoff	Y	N	N	N	N	Y
pd_java_classpath	Y	N	N	N	N	Y
pd_java_libpath	Y	N	N	N	N	Y
pd_java_option	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_java_routine_stack_size	Y	N	N	N	N	Y
pd_java_runtimepath	Y	N	N	N	N	Y
pd_java_stdout_file	Y	N	N	N	N	Y
pd_jp1_event_level	Y	N	N	N	N	Y
pd_jp1_event_msg_out	Y	N	N	N	N	Y
pd_jp1_use	Y	N	N	N	N	Y
pd_key_resource_type	Y	N	N	N	N	Y
pd_large_file_use	Y	N	N	N	N	Y
pd_lck_deadlock_check	Y	N	N	N	N	Y
pd_lck_deadlock_check_interval	Y	N	N	N	N	Y
pd_lck_deadlock_info	Y	N	N	N	N	Y
pd_lck_hash_entry	Y	N	N	N	N	Y
pd_lck_pool_partition	Y	N	N	N	N	Y
pd_lck_pool_size	Y	N	N	N	N	Y
pd_lck_queue_limit	Y	N	N	N	N	Y
pd_lck_release_detect	Y	N	N	N	N	Y
pd_lck_release_detect_interval	Y	N	N	N	N	Y
pd_lck_until_disconnect_cnt	Y	N	N	N	N	Y
pd_lck_wait_timeout	Y	N	N	N	N	Y
pd_leap_second	Y	N	N	N	N	Y
pd_list_initialize_timing	Y	N	N	N	N	N
pd_listen_socket_bufset	Y	N	N	N	N	Y
pd_lock_uncommited_delete_data	Y	N	N	N	N	Y
pd_log_auto_expand_size	Y	N	N	N	N	Y
pd_log_auto_unload_path	Y	N	N	N	N	Y
pd_log_auto_unload_restart	Y	N	N	N	N	Y
pd_log_dual	Y	N	N	N	N	Y
pd_log_max_data_size	Y	N	N	N	N	Y
pd_log_rec_leng	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_log_remain_space_check	Y	N	N	N	N	N
pd_log_rerun_reserved_file_open	Y	N	N	N	N	Y
pd_log_rerun_swap	Y	N	N	N	N	Y
pd_log_rollback_buff_count	Y	N	N	N	N	Y
pd_log_rpl_no_standby_file_opr	Y	N	N	N	N	Y
pd_log_sdinterval	Y	N	N	N	N	Y
pd_log_singleoperation	Y	N	N	N	N	Y
pd_log_swap_timeout	Y	N	N	N	N	Y
pd_log_unload_check	Y	N	N	N	N	Y
pd_log_write_buff_count	Y	N	N	N	N	Y
pd_master_file_name	Y	N	N	N	N	Y
pd_max_access_tables	Y	N	N	N	N	Y
pd_max_access_tables_wrn_pnt	Y	N	N	N	N	Y
pd_max_add_dbbuff_no	Y	N	N	N	N	Y
pd_max_add_dbbuff_shm_no	Y	N	N	N	N	Y
pd_max_ard_process	Y	N	N	N	N	N
pd_max_bes_process	Y	N	N	N	N	Y
pd_max_commit_write_reclaim_no	Y	N	N	N	N	Y
pd_max_dbbuff_shm_no	Y	N	N	N	N	Y
pd_max_dic_process	Y	N	N	N	N	Y
pd_max_file_no	Y	N	N	N	N	Y
pd_max_file_no_wrn_pnt	Y	N	N	N	N	Y
pd_max_list_users	Y	N	N	N	N	Y
pd_max_list_count	Y	N	N	N	N	Y
pd_max_list_users_wrn_pnt	Y	N	N	N	N	Y
pd_max_list_count_wrn_pnt	Y	N	N	N	N	Y
pd_max_open_holdable_cursors	Y	N	N	N	N	Y
pd_max_rdarea_no	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_max_rdarea_no_wrn_pnt	Y	N	N	N	N	Y
pd_max_recover_process	Y	N	N	N	N	Y
pd_max_resident_rdarea_no	Y	N	N	N	N	Y
pd_max_resident_rdarea_shm_no	Y	N	N	N	N	Y
pd_max_server_process	Y	N	N	N	N	Y
pd_max_temporary_object_no	Y	N	N	N	N	Y
pd_max_tmp_table_rdarea_no	Y	N	N	N	N	Y
pd_max_users	Y	N	N	N	N	Y
pd_max_users_wrn_pnt	Y	N	N	N	N	Y
pd_mlg_file_size	Y	N	N	N	N	Y
pd_mlg_msg_log_unit	Y	N	N	N	N	Y
pd_mlg_port	Y	N	N	N	N	Y
pd_mode_conf	Y	N	N	N	N	Y
pd_module_trace_max	Y	N	N	N	N	Y
pd_module_trace_timer_level	Y	N	N	N	N	Y
pd_name_fixed_port_lookup	Y	N	N	N	N	Y
pd_name_port	Y	N	N	N	N	Y
pd_node_name	Y	N	N	N	N	N
pd_non_floatable_bes	Y	N	N	N	N	N
pd_nowait_scan_option	Y	N	N	N	N	Y
pd_ntfs_cache_disable	Y	N	N	N	N	Y
pd_oltp_holdcr	Y	N	N	N	N	Y
pd_optimize_level	Y	N	N	N	N	Y
pd_overflow_suppress	Y	N	N	N	N	Y
pd_pageaccess_mode	Y	N	N	N	N	Y
pd_plugin_ixmk_dir	Y	N	N	N	N	N
pd_prf_file_count	Y	N	N	N	N	Y
pd_prf_file_size	Y	N	N	N	N	Y
pd_prf_level	Y	N	N	N	N	Y
pd_prf_trace	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_process_count	Y	N	N	N	N	Y
pd_process_desktopheap_size	Y	N	N	N	N	N
pd_process_terminator	Y	N	N	N	N	Y
pd_process_terminator_max	Y	N	N	N	N	Y
pd_pth_trace_max	Y	N	N	N	N	Y
pd_queue_watch_time	Y	N	N	N	N	Y
pd_queue_watch_timeover_action	Y	N	N	N	N	Y
pd_rdarea_expand_format	Y	N	N	N	N	Y
pd_rdarea_extension_timing	Y	N	N	N	N	Y
pd_rdarea_list_no_wrn_pnt	Y	N	N	N	N	Y
pd_rdarea_open_attribute	Y	N	N	N	N	Y
pd_rdarea_open_attribute_use	Y	N	N	N	N	Y
pd_rdarea_warning_point	Y	N	N	N	N	Y
pd_rdarea_warning_point_msgout	Y	N	N	N	N	Y
pd_redo_allpage_put	Y	N	N	N	N	Y
pd_reduced_check_time	Y	N	N	N	N	Y
pd_registered_port	Y	Y	Y	Y	Y	Y
pd_registered_port_check	Y	Y	Y	Y	Y	Y
pd_registered_port_level	Y	Y	Y	Y	Y	Y
pd_registry_cache_size	Y	N	N	N	N	Y
pd_rorg_predict	Y	N	N	N	N	Y
pd_routine_def_cache_size	Y	N	N	N	N	Y
pd_rpc_bind_loopback_address	Y	N	N	N	N	Y
pd_rpc_trace	Y	N	N	N	N	Y
pd_rpc_trace_name	Y	N	N	N	N	Y
pd_rpc_trace_size	Y	N	N	N	N	Y
pd_rpl_func_control	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
pd_rpl_hdepath	Y	N	N	N	N	N
pd_rpl_init_start	Y	N	N	N	N	Y
pd_rpl_reflect_mode	Y	N	N	N	N	Y
pd_scd_port	Y	N	N	N	N	Y
pd_sds_shmpool_size	Y	N	N	N	N	Y
pd_security_host_group	Y	N	N	N	Y	Y
pd_server_cleanup_interval	Y	N	N	N	N	Y
pd_server_entry_queue	Y	N	N	N	N	Y
pd_service_port	Y	N	N	N	N	Y
pd_shared_memory_report	Y	N	N	N	N	Y
pd_shared_rdarea_use	Y	N	N	N	N	Y
pd_shmpool_attribute	Y	N	N	N	N	Y
pd_shmpool_control	Y	N	N	N	N	Y
pd_space_level	Y	N	N	N	N	Y
pd_spd_assurance_count	Y	N	N	N	N	Y
pd_spd_assurance_msg	Y	N	N	N	N	Y
pd_spd_dual	Y	N	N	N	N	Y
pd_spd_max_data_size	Y	N	N	N	N	Y
pd_spd_reduced_mode	Y	N	N	N	N	Y
pd_spd_reserved_file_auto_open	Y	N	N	N	N	Y
pd_spd_syncpoint_skip_limit	Y	N	N	N	N	Y
pd_spool_cleanup	Y	N	N	N	N	N
pd_spool_cleanup_interval	Y	N	N	N	N	N
pd_spool_cleanup_interval_level	Y	N	N	N	N	N
pd_spool_cleanup_level	Y	N	N	N	N	N
pd_sql_command_exec_users	Y	N	N	N	N	Y
pd_sql_dec_op_maxprec	Y	N	N	N	N	Y
pd_sql_mode	Y	N	N	N	N	Y
pd_sql_object_cache_size	Y	N	N	N	N	Y
pd_sql_send_buff_size	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
`pd_sql_simple_comment_use`	Y	N	N	N	N	Y
`pd_sqlobject_stat_timing`	Y	N	N	N	N	Y
`pd_start_level`	Y	N	N	N	N	Y
`pd_start_skip_unit`	Y	N	N	N	N	Y
`pd_start_time_out`	Y	N	N	N	N	Y
`pd_statistics`	Y	N	N	N	N	Y
`pd_stj_file_size`	Y	N	N	N	N	Y
`pd_stj_buff_size`	Y	N	N	N	N	Y
`pd_standard_sqlstate`	Y	N	N	N	N	Y
`pd_sts_file_name_1-7`	Y	Y	Y	Y	N	N
`pd_sts_initial_error`	Y	N	N	N	N	Y
`pd_sts_last_active_file`	Y	N	N	N	N	N
`pd_sts_last_active_side`	Y	N	N	N	N	N
`pd_sts_singleoperation`	Y	N	N	N	N	Y
`pd_substr_length`	Y	N	N	N	N	Y
`pd_sysdef_default_option`	Y	N	N	N	N	Y
`pd_syssts_file_name_1-7`	Y	N	N	N	N	Y
`pd_syssts_initial_error`	Y	N	N	N	N	Y
`pd_syssts_last_active_file`	Y	N	N	N	N	Y
`pd_syssts_last_active_side`	Y	N	N	N	N	Y
`pd_syssts_singleoperation`	Y	N	N	N	N	Y
`pd_system_complete_wait_time`	Y	N	N	N	N	Y
`pd_system_dbsync_point`	Y	N	N	N	N	Y
`pd_system_id`	Y	N	N	N	N	Y
`pd_table_def_cache_size`	Y	N	N	N	N	Y
`pd_tcp_inet_bufsize`	Y	N	N	N	N	Y
`pd_thdlock_pipe_retry_interval`	Y	N	N	N	N	Y
`pd_thdlock_retry_time`	Y	N	N	N	N	Y
`pd_thdlock_sleep_func`	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
`pd_thdlock_wakeup_lock`	Y	N	N	N	N	Y
`pd_thdspnlk_spn_count`	Y	N	N	N	N	Y
`pd_thread_max_stack_size`	Y	N	N	N	N	Y
`pd_thread_stack_expand_size`	Y	N	N	N	N	Y
`pd_tmp_directory`	Y	Y	Y	N	N	N
`pd_tmp_table_initialize_timing`	Y	N	N	N	N	Y
`pd_trn_commit_optimize`	Y	N	N	N	N	Y
`pd_trn_port`	Y	N	N	N	N	Y
`pd_trn_rcvmsg_store_buflen`	Y	N	N	N	N	Y
`pd_trn_rerun_branch_auto_decide`	Y	N	N	N	N	Y
`pd_trn_rollback_msg_interval`	Y	N	N	N	N	Y
`pd_trn_rollback_watch_time`	Y	N	N	N	N	Y
`pd_trn_send_decision_interval`	Y	N	N	N	N	Y
`pd_trn_send_decision_intval_sec`	Y	N	N	N	N	Y
`pd_trn_send_decision_retry_time`	Y	N	N	N	N	Y
`pd_trn_watch_time`	Y	N	N	N	N	Y
`pd_type_def_cache_size`	Y	N	N	N	N	Y
`pd_uap_exerror_log_dir`	Y	N	N	N	N	Y
`pd_uap_exerror_log_param_size`	Y	N	N	N	N	Y
`pd_uap_exerror_log_size`	Y	N	N	N	N	Y
`pd_uap_exerror_log_use`	Y	N	N	N	N	Y
`pd_unit_id`	Y	N	N	N	N	Y
`pd_utl_buff_size`	Y	N	N	N	N	Y
`pd_utl_exec_mode`	Y	N	N	N	N	Y
`pd_utl_exec_time`	Y	N	N	N	N	Y
`pd_view_def_cache_size`	Y	N	N	N	N	Y
`pd_watch_pc_client_time`	Y	N	N	N	N	Y
`pd_watch_resource`	Y	N	N	N	N	Y

Operand	Syntax check	File check	Access privilege check	Duplicate specification check	Host name check	Check among server machines
`pd_watch_time`	Y	N	N	N	N	Y
`pd_work_buff_expand_limit`	Y	N	N	N	N	Y
`pd_work_buff_mode`	Y	N	N	N	N	Y
`pd_work_buff_size`	Y	N	N	N	N	Y
`pd_work_table_option`	Y	N	N	N	N	Y
`pdaudload`	Y	N	N	N	N	Y
`pdbuffer`	Y	N	N	N	N	Y
`pdcltgrp`	Y	N	N	N	N	Y
`pdhagroup`	Y	N	N	Y	N	Y
`pdhibegin`	Y	N	N	N	N	Y
`pdhubopt`	Y	N	N	N	N	N
`pdlogadfg -d spd`	Y	Y	Y	Y	N	N
`pdlogadfg -d sys`	Y	Y	Y	Y	N	N
`pdlogadpf -d spd`	Y	Y	Y	Y	N	N
`pdlogadpf -d sys`	Y	Y	Y	Y	N	N
`pdmlgput`	Y	N	N	N	N	Y
`pdplgprm`	Y	N	N	N	N	N
`pdplugin`	Y	N	N	N	N	Y
`pdstart`	Y	N	N	N	Y	Y
`pdstbegin`	Y	N	N	N	N	Y
`pdunit`	Y	N	N	N	Y	Y
`pdwork`	Y	N	N	N	N	N
`pdwork_wrn_pnt`	Y	N	N	N	N	Y

Y: Operand is checked.

N: Operand cannot be checked.

Syntax check:

Checks whether the operand's syntax is correct.

File check:

Checks for the presence of the system log file, synchronization point dump file, and status file. File check is not performed if the `-n` option is specified for the `pdconfchk` command.

Access privilege check:

Checks whether the HiRDB administrator has file access privileges.

Access privilege check is not performed if the `-n` option is specified for the `pdconfchk` command.

Checks whether the user has access privileges for the directory specified in the `pd_tmp_directory` operand. If neither `pd_tmp_directory` nor the `TMP` environment variable is specified, access privileges are not checked.

Duplicate specification check:

Checks whether the system log files, synchronization point dump files, or status files are duplicated.

Host name check:

Checks whether the host name is described in the hosts file.

Check among server machines (applicable only to HiRDB/Parallel Server):

Checks for compatibility among server machines using the server machine of the system manager as the reference.

F. List of Operands That Can Be Specified When Using the Standby-less System Switchover (Effects Distributed) Facility (Unit Control Information Definition)

When using the standby-less system switchover (effects distributed) facility, you can specify the following operands in the unit control information definition. You cannot specify any other operands in the unit control information definition. If any other operand is specified, HiRDB cannot start (the message `KFPS05618-E` is output).

- `pd_aud_file_name`#
- `pd_down_watch_proc`
- `pd_ha_acttype`
- `pd_ha_agent`
- `pd_ha_max_act_guest_servers`
- `pd_ha_max_server_process`
- `pd_ha_process_count`
- `pd_ha_resource_act_wait_time`
- `pd_ha_unit`
- `pd_hostname`
- `pd_ipc_conn_nblock_time`
- `pd_lck_deadlock_check`
- `pd_lck_deadlock_check_interval`
- `pd_prf_file_count`
- `pd_prf_file_size`
- `pd_prf_level`
- `pd_process_desktopheap_size`
- `pd_registered_port`#
- `pd_rpc_trace_name`#
- `pd_rpl_hdepath`
- `pd_service_port`
- `pd_syssts_file_name_1` to `7`
- `pd_syssts_initial_error`
- `pd_syssts_last_active_file`
- `pd_syssts_last_active_side`
- `pd_syssts_singleoperation`
- `pd_tmp_directory`
- `pd_trn_rcvmsg_store_buflen`
- `pd_unit_id`

#: When multiple HiRDB units are allocated to the same server machine, the specification must be different for each unit.

Index