

Informix Product Family
Informix
Version 11.70

IBM Informix Administrator's Reference



Informix Product Family
Informix
Version 11.70

IBM Informix Administrator's Reference



Note

Before using this information and the product it supports, read the information in "Notices" on page H-1.

Edition

This edition replaces SC27-3530-02.

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corporation 1996, 2011.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Introduction	xxiii
In This Introduction	xxiii
About This Publication	xxiii
Types of Users.	xxiii
Software Dependencies	xxiii
Assumptions About Your Locale.	xxiv
Demonstration Databases	xxiv
What's New in Administrator's Reference for Informix database server, Version 11.70	xxiv
Example code conventions	xxxiv
Additional documentation	xxxiv
Compliance with industry standards	xxxv
Syntax diagrams	xxxv
How to read a command-line syntax diagram	xxxvi
Keywords and punctuation	xxxvii
Identifiers and names.	xxxvii
How to provide documentation feedback	xxxviii

Part 1. Configuring and Monitoring Informix

Chapter 1. Database configuration parameters	1-1
onconfig File Conventions	1-1
Creating the onconfig File	1-2
Displaying the Settings in the onconfig File	1-2
onconfig Portal: Configuration parameters by functional category	1-3
Database configuration parameter reference	1-24
ADMIN_MODE_USERS Configuration Parameter	1-25
ADMIN_USER_MODE_WITH_DBSA Configuration Parameter	1-26
ALARMPROGRAM configuration parameter	1-27
ALLOW_NEWLINE Configuration Parameter	1-28
ALRM_ALL_EVENTS configuration parameter	1-29
AUTO_AIOVPS Configuration Parameter.	1-29
AUTO_CKPTS Configuration Parameter	1-30
AUTO_LRU_TUNING Configuration Parameter	1-30
AUTO_READAHEAD configuration parameter.	1-31
AUTO_REPREPARE configuration parameter	1-32
AUTO_STAT_MODE configuration parameter	1-33
BATCHEDREAD_TABLE configuration parameter.	1-34
BATCHEDREAD_INDEX configuration parameter.	1-34
BLOCKTIMEOUT configuration parameter	1-35
BTSCANNER Configuration Parameter	1-35
BUFFERPOOL configuration parameter	1-36
CHECKALLDOMAINSFORUSER configuration parameter	1-40
CKPTINTVL configuration parameter	1-41
CLEANERS Configuration Parameter	1-41
CONSOLE Configuration Parameter	1-42
CONVERSION_GUARD configuration parameter	1-42
DATASKIP Configuration Parameter	1-43
DBCREATE_PERMISSION Configuration Parameter	1-44
DB_LIBRARY_PATH Configuration Parameter	1-44
DBSERVERALIASES configuration parameter	1-45
DBSERVERNAME configuration parameter	1-47
DBSPACETEMP configuration parameter	1-48
Use Hash Join Overflow and DBSPACETEMP	1-50
DD_HASHMAX configuration parameter.	1-50
DD_HASHSIZE Configuration Parameter.	1-51

DEADLOCK_TIMEOUT configuration parameter	1-51
DEF_TABLE_LOCKMODE configuration parameter	1-52
DEFAULTESCCHAR configuration parameter	1-53
DELAY_APPLY Configuration Parameter	1-53
DIRECT_IO Configuration Parameter (UNIX)	1-54
DIRECTIVES configuration parameter	1-55
DISABLE_B162428_XA_FIX Configuration Parameter	1-56
DRDA_COMMBUFFSIZE Configuration Parameter	1-57
DRAUTO Configuration Parameter	1-57
DRIDXAUTO Configuration Parameter	1-59
DRINTERVAL configuration parameter	1-59
DRLOSTFOUND Configuration Parameter	1-60
DRTIMEOUT configuration parameter	1-60
DS_HASHSIZE Configuration Parameter	1-61
DS_MAX_QUERIES Configuration Parameter	1-61
DS_MAX_SCANS Configuration Parameter	1-62
DS_NONPDQ_QUERY_MEM configuration parameter	1-63
DS_POOLSIZE Configuration Parameter	1-64
DS_TOTAL_MEMORY configuration parameter.	1-64
Algorithm for DS_TOTAL_MEMORY	1-65
AUTO_STAT_MODE configuration parameter	1-66
DUMPCNT Configuration Parameter (UNIX)	1-66
DUMPCORE Configuration Parameter (UNIX)	1-67
DUMPDIR Configuration Parameter	1-67
DUMPGCORE Configuration Parameter (UNIX)	1-68
DUMPSHMEM Configuration Parameter (UNIX)	1-68
DYNAMIC_LOGS Configuration Parameter	1-69
EILSEQ_COMPAT_MODE configuration parameter	1-70
ENABLE_SNAPSHOT_COPY Configuration Parameter	1-70
ENCRYPT_CIPHERS Configuration Parameter	1-71
ENCRYPT_HDR Configuration Parameter	1-72
ENCRYPT_MAC Configuration Parameter	1-73
ENCRYPT_MACFILE configuration parameter	1-74
ENCRYPT_SMX Configuration Parameter	1-75
ENCRYPT_SWITCH Configuration Parameter	1-75
EXPLAIN_STAT Configuration Parameter	1-76
EXT_DIRECTIVES Configuration Parameter	1-76
EXTSHMADD Configuration Parameter	1-77
FAILOVER_CALLBACK Configuration Parameter	1-77
FAILOVER_TX_TIMEOUT Configuration Parameter	1-78
FASTPOLL Configuration Parameter	1-79
FILLFACTOR configuration parameter.	1-79
FULL_DISK_INIT Configuration Parameter	1-80
GSKIT_VERSION configuration parameter	1-80
HA_ALIAS Configuration Parameter	1-81
HETERO_COMMIT Configuration Parameter	1-81
IFX_EXTEND_ROLE Configuration Parameter	1-82
IFX_FOLDVIEW configuration parameter.	1-82
LIMITNUMSESSIONS configuration parameter.	1-83
LISTEN_TIMEOUT Configuration Parameter	1-84
LOCKS Configuration Parameter	1-84
LOGBUFF configuration parameter.	1-85
LOGFILES configuration parameter.	1-86
LOG_INDEX_BUILDS Configuration Parameter	1-87
LOG_STAGING_DIR Configuration Parameter	1-87
LOGSIZE configuration parameter	1-88
LOGSIZE for smart large objects.	1-89
LOW_MEMORY_MGR configuration parameter	1-89
LOW_MEMORY_RESERVE configuration parameter	1-90
LTXEHWM Configuration Parameter	1-91
LTXHWM Configuration Parameter.	1-91

MAX_FILL_DATA_PAGES Configuration Parameter	1-92
MAX_INCOMPLETE_CONNECTIONS Configuration Parameter	1-93
MAX_PDQPRIORITY Configuration Parameter	1-94
MIRROR Configuration Parameter	1-95
MIRROROFFSET Configuration Parameter	1-95
MIRRORPATH Configuration Parameter	1-95
MSG_DATE configuration parameter	1-96
MSGPATH configuration parameter	1-96
MULTIPROCESSOR Configuration Parameter	1-97
NET_IO_TIMEOUT_ALARM configuration parameter	1-97
NETTYPE Configuration Parameter.	1-98
NS_CACHE configuration parameter	1-101
NUMFDSERVERS configuration parameter	1-102
OFF_RECVRY_THREADS configuration parameter	1-103
ON_RECVRY_THREADS Configuration Parameter	1-103
ONDBSPACEDOWN configuration parameter	1-104
Database Server Behavior When ONDBSPACEDOWN Does Not Apply	1-105
ONLIDX_MAXMEM Configuration Parameter.	1-105
OPCACHEMAX Configuration Parameter (UNIX)	1-106
OPTCOMPIND configuration parameter.	1-106
OPT_GOAL Configuration Parameter.	1-107
PC_HASHSIZE Configuration Parameter	1-108
PC_POOLSIZ Configuration Parameter.	1-109
PHYSBUFF Configuration Parameter	1-109
PHYSFILE Configuration Parameter	1-110
PLOG_OVERFLOW_PATH Configuration Parameter	1-110
PN_STAGELOB_THRESHOLD configuration parameter	1-111
PRELOAD_DLL_FILE Configuration Parameter	1-112
QSTATS Configuration Parameter	1-112
RA_PAGES configuration parameter	1-113
REMOTE_SERVER_CFG configuration parameter.	1-113
REMOTE_USERS_CFG configuration parameter	1-114
RESIDENT configuration parameter	1-115
RESTARTABLE_RESTORE Configuration Parameter	1-116
RESTORE_POINT_DIR configuration parameter	1-117
ROOTNAME Configuration Parameter	1-117
ROOTOFFSET Configuration Parameter	1-117
ROOTPATH Configuration Parameter	1-118
ROOTSIZE configuration parameter	1-118
RSS_FLOW_CONTROL Configuration Parameter.	1-119
RTO_SERVER_RESTART configuration parameter	1-120
S6_USE_REMOTE_SERVER_CFG configuration parameter.	1-121
SBSPECNAME Configuration Parameter	1-121
SBSPECTEMP Configuration Parameter	1-123
SDS_ENABLE Configuration Parameter	1-123
SDS_LOGCHECK configuration parameter	1-124
SDS_PAGING Configuration Parameter	1-125
SDS_TEMPDBS Configuration Parameter	1-125
SDS_TIMEOUT Configuration Parameter	1-127
SECURITY_LOCALCONNECTION Configuration Parameter.	1-127
SEQ_CACHE_SIZE configuration parameter	1-127
SERVERNUM configuration parameter	1-128
SHMADD configuration parameter	1-128
SHMBASE Configuration Parameter	1-129
SHMNOACCESS Configuration Parameter	1-130
SHMTOTAL configuration parameter.	1-130
SHMVIRT_ALLOCSEG configuration parameter	1-131
SHMVIRTSIZE configuration parameter	1-132
SINGLE_CPU_VP configuration parameter	1-134
VPCLASS Values and the SINGLE_CPU_VP Configuration Parameter	1-134
SMX_COMPRESS Configuration Parameter.	1-135

SP_AUTOEXPAND configuration parameter	1-136
SP_THRESHOLD Configuration Parameter	1-136
SP_WAITTIME Configuration Parameter	1-137
SQL_LOGICAL_CHAR Configuration Parameter	1-138
SQLTRACE Configuration Parameter	1-139
SSL_KEYSTORE_LABEL Configuration Parameter	1-140
STACKSIZE configuration parameter	1-141
STAGEBLOB Configuration Parameter	1-141
STATCHANGE configuration parameter.	1-142
STMT_CACHE Configuration Parameter	1-142
STMT_CACHE_HITS Configuration Parameter	1-143
STMT_CACHE_NOLIMIT Configuration Parameter	1-144
STMT_CACHE_NUMPOOL Configuration Parameter	1-145
STMT_CACHE_SIZE Configuration Parameter	1-145
STOP_APPLY Configuration Parameter	1-145
STORAGE_FULL_ALARM Configuration Parameter.	1-147
SYSEALARMPROGRAM configuration parameter	1-147
SYSSBSPACENAME Configuration Parameter	1-148
TBLSPACE_STATS configuration parameter	1-149
TBLTBLFIRST configuration parameter	1-150
TBLTBLNEXT configuration parameter	1-150
TEMPTAB_NOLOG configuration parameter	1-151
TXTIMEOUT configuration parameter	1-151
UNSECURE_ONSTAT Configuration Parameter	1-152
UPDATABLE_SECONDARY Configuration Parameter	1-152
USELASTCOMMITTED Configuration Parameter	1-153
USEOSTIME Configuration Parameter	1-155
USTLOW_SAMPLE configuration parameter	1-155
VP_MEMORY_CACHE_KB Configuration Parameter	1-156
VPCLASS configuration parameter	1-156
WSTATS configuration parameter	1-163
USERMAPPING configuration parameter	1-163

Chapter 2. The sysmaster database 2-1

The sysmaster Database	2-1
The buildsmi Script	2-1
The bldutil.sh Script	2-2
The System-Monitoring Interface	2-2
Understanding the SMI Tables	2-2
Accessing SMI tables	2-3
The System-Monitoring Interface Tables.	2-4
The sysutils Tables.	2-6
sysadinfo	2-7
sysaudit	2-7
syschkio	2-8
syscheckpoint	2-8
syschunks	2-9
syscluster	2-10
syscsm.	2-11
syscsmsla.	2-12
syscsmstab.	2-12
syscompdicts_full.	2-12
sysconfig	2-14
sysdatabases	2-14
sysdbslocale	2-15
sysdbspaces	2-15
sysdri.	2-16
sysdual	2-17
sysenv	2-17
sysenvses	2-17
sysextents	2-17

sysextspaces	2-18
sysfeatures	2-18
syssha_lagtime Table	2-19
syssha_type	2-20
syssha_workload	2-20
sysipl	2-21
syslocks	2-21
syslogs	2-22
syslogfil table	2-23
sysmgminfo	2-23
sysnetclienttype	2-24
sysnetglobal	2-24
sysnetworkio table	2-25
sysonlinelog	2-25
sysprofile	2-26
sysproxyagents	2-27
sysproxydistributors	2-28
sysproxysessions table	2-28
sysproxytxnops table	2-29
sysproxytxns table	2-29
sysptprof table.	2-30
sysrepevtreg table	2-30
sysrepstats table	2-31
sysrsslog.	2-34
sysscblst	2-34
sysseappinfo	2-34
sysseprof	2-35
syssessions	2-35
sysstmx	2-37
sysstmxses	2-37
syssexplain table	2-37
syssqltrace	2-38
syssqltrace_info	2-40
syssqltrace_iter	2-40
syssrcrs	2-40
syssrcsds	2-41
systabnames	2-41
systhreads	2-42
sysstrgrss.	2-42
sysstrgsds	2-43
sysvpprof	2-43
The SMI Tables Map.	2-44
Information from onstat in the SMI Tables	2-45
Chapter 3. The sysadmin Database	3-1
The Scheduler Tables	3-1
The ph_task Table	3-2
The ph_run Table	3-4
The ph_group Table	3-5
The ph_alert Table	3-5
The ph_threshold Table	3-7
Results Tables	3-8
The command_history table.	3-8
The storagepool table.	3-9
Chapter 4. Disk Structures and Storage	4-1
Dbpace Structure and Storage.	4-1
Structure of the Root Dbpace	4-1
Reserved Pages	4-2
Structure of a Regular Dbpace	4-2

Structure of the Chunk Free-List Page	4-4
Structure of the Tblspace Tblspace	4-4
Structure of the Database Tblspace	4-7
Structure and Allocation of an Extent	4-7
Structure and Storage of a Dbspace Page	4-12
Structure of Fragmented Tables	4-15
Structure of B-Tree Index Pages	4-16
Structure of R-Tree Index Pages	4-21
Storage of Simple Large Objects	4-21
Structure of a Blobspace	4-21
Structure of a Dbspace Blobpage.	4-22
Simple-Large-Object Storage and the Descriptor	4-22
Blobspace Page Types	4-23
Structure of a Blobspace Blobpage	4-23
Sbspace Structure.	4-24
Structure of the metadata area	4-25
Sbpage Structure	4-26
Multiple Chunk Sbspace	4-26
Time Stamps	4-27
Database and Table Creation: What Happens on Disk.	4-27
Database Creation	4-27
Table Creation	4-28

Chapter 5. Interpreting Logical-Log Records 5-1

About Logical-Log Records	5-1
Transactions That Drop a Table or Index	5-1
Transactions That Are Rolled Back	5-1
Checkpoints with Active Transactions	5-2
Distributed Transactions	5-2
Logical-Log Record Structure	5-2
Logical-Log Record Header.	5-2
Logical-Log Record Types and Additional Columns.	5-3
Log Record Types for Smart Large Objects	5-14

Part 2. Administrative Utilities

Chapter 6. Overview of Utilities 6-1

Obtaining Utility Version Information	6-1
IBM Informix Server Administrator	6-2

Chapter 7. The genoncfg Utility 7-1

Chapter 8. The oncheck Utility 8-1

oncheck Check-and-Repair	8-1
What Does Each Option Do?	8-1
Using the -y Option to Perform Repairs.	8-2
Repairing Indexes in Sbspaces and External Spaces	8-3
Locking and oncheck	8-3
oncheck syntax	8-3
oncheck -cc and -pc: Check system catalog tables.	8-8
oncheck -cd and oncheck -cD commands: Check pages	8-8
oncheck -ce, -pe: Check the chunk-free list	8-10
oncheck -ci and -cI : Check index node links.	8-10
oncheck -cr and -cR: Check reserved pages	8-12
oncheck -cs, -cS, -ps, -pS: Check and display sbspaces	8-13
oncheck -pB: Display blobspace statistics	8-13
oncheck -pd and pD: Display rows in hexadecimal format	8-13
oncheck -pk, -pK, -pl, -pL: Display index information.	8-15
oncheck -pp and -pP: Display the contents of a logical page	8-16

oncheck -pr and pR: Display reserved-page information	8-18
oncheck -pt and -pT: Display tblspaces for a Table or Fragment	8-19
Turn On Locking with -x	8-21
Send Special Arguments to the Access Method with -u	8-22
Return Codes on Exit	8-22
Chapter 9. The onclean Utility	9-1
The onshutdown script	9-2
Chapter 10. The oncmsm utility	10-1
Service Level Agreement Examples	10-11
Failover Configuration Examples	10-15
Proxy Mode and Redirect Mode	10-16
Chapter 11. The ondblog Utility	11-1
ondblog: Change Logging Mode.	11-1
ondblog Syntax	11-1
Chapter 12. The oninit utility	12-1
Return codes for the oninit utility	12-4
Chapter 13. The onlog Utility	13-1
Chapter 14. The onmode utility	14-1
onmode Syntax	14-1
onmode -a: Add a shared-memory segment	14-2
onmode -BC: Allow large chunk mode.	14-3
onmode -c: Force a checkpoint	14-4
onmode -C: Control the B-tree scanner.	14-5
onmode -cache surrogates: Cache the allowed.surrogates file	14-6
onmode -d: Set data-replication types	14-6
Using the -d standard Option.	14-7
Using the -d primary dbservername Option	14-7
Using the -d secondary dbservername option	14-8
onmode -d: Set High Availability server characteristics	14-8
onmode -d command: Replicate an index with data-replication	14-10
onmode -D, -M, -Q, -S: Change decision-support parameters	14-11
onmode -e: Change usage of the SQL statement cache	14-13
onmode -F: Free unused memory segments.	14-13
onmode -I: Control diagnostics collection	14-14
onmode -k, -m, -s, -u, -j: Change database server mode.	14-15
Taking the Database Server to Offline Mode with the -k Option	14-16
Bringing the Database Server Online with the -m Option	14-16
Shutting Down the Database Server Gracefully with the -s Option	14-16
Shutting Down the Database Server Immediately with the -u Option	14-16
Changing the Database Server to Administration Mode with the -j Option	14-17
Changing Database Server Mode with ON-Monitor (UNIX)	14-17
onmode -l: Switch the logical-log file	14-18
onmode -n, -r: Change shared-memory residency.	14-18
onmode -O: Override ONDBSPACEDOWN WAIT mode	14-19
onmode -p: Add or remove virtual processors.	14-19
Adding and Dropping Virtual Processors	14-21
Dropping Virtual Processors Automatically	14-21
Monitoring poll threads with the onstat utility.	14-22
onmode -P: Start, stop, or restart a listen thread dynamically	14-22
onmode -R: Regenerate .infos File	14-23
onmode -W: Change settings for the SQL statement cache	14-24
SQL statement cache examples	14-24
onmode -wf, -wm: Dynamically change certain configuration parameters.	14-25

onmode -wm: Change LRU tuning status	14-27
onmode -Y: Dynamically change SET EXPLAIN	14-28
onmode -z: Kill a database server session	14-29
onmode -Z: Kill a distributed transaction	14-29

Chapter 15. The ON-Monitor Utility 15-1

Using ON-Monitor (UNIX)	15-1
Navigating ON-Monitor and Using Help	15-1
Executing Shell Commands Within ON-Monitor	15-2
ON-Monitor Screen Options	15-2
Setting Configuration Parameters in ON-Monitor	15-3

Chapter 16. The onparams Utility 16-1

onparams Syntax	16-1
onparams -a -d <i>dbspace</i> : Add a logical-log file	16-2
onparams -d -l <i>lognum</i> : Drop a logical-log file	16-2
onparams -p: Change physical-log parameters	16-3
Backing Up After You Change the Physical-Log Size or Location	16-4
Changing the Size of the Physical Log and Using Non-Default Page Sizes	16-4
onparams -b: Add a new buffer pool	16-4
Examples of onparams Commands	16-6

Chapter 17. The onpassword utility. 17-1

Configuring secure connections using a password file	17-3
--	------

Chapter 18. The ifxclone utility 18-1

Chapter 19. The onspaces Utility. 19-1

onspaces Syntax	19-1
onspaces -a: Add a chunk to a dbspace or blobspace	19-2
onspaces -a: Add a chunk to an sbpace	19-3
onspaces -c -b: Create a blobspace	19-5
onspaces -c -d: Create a dbspace.	19-7
Creating a Temporary Dbpace with the -t Option	19-10
Specifying First and Next Extent Size for the tblspace tblspace	19-10
Specifying a Non-Default Page Size with the Same Size as the Buffer Pool	19-10
onspaces -c -S: Create an sbpace	19-11
Creating a Temporary Sbpce with the -t Option	19-12
Creating an Sbpce with the -Df option.	19-12
Changing the -Df Settings	19-15
Using the onspaces -g option	19-16
onspaces -c -x: Create an extspace	19-16
onspaces -ch: Change sbpace default specifications	19-17
onspaces -cl: Clean up stray smart large objects in sbspaces	19-18
onspaces -d: Drop a chunk in a dbspace, blobpace, or sbpace	19-18
onspaces -d: Drop a blobpace, dbspace, extspace, or sbpace.	19-20
onspaces -f: Specify DATASKIP parameter	19-21
onspaces -m: Start mirroring.	19-22
Using a File to Specify Chunk-Location Information with the -f Option	19-23
onspaces -r: Stop mirroring	19-24
onspaces -ren: Rename a dbspace, blobpace, sbpace, or extspace	19-24
Renaming a Dbpace, Blobpace, Sbpce, or Extspace when Enterprise Replication Is Active	19-25
Performing an Archive after Renaming a Space	19-26
onspaces -s: Change status of a mirrored chunk	19-26
Avoid overwriting a chunk	19-27

Chapter 20. The onstat utility 20-1

onstat Portal: onstat Utility Commands Sorted by Functional Category	20-2
Monitor the database server status.	20-18

onstat Command Syntax	20-19
onstat command: Equivalent to the onstat -pu command	20-23
onstat - command: Print output header	20-23
Subheader printed with the onstat - command	20-24
onstat - command: Print onstat options and functions	20-25
Running onstat Commands on a Shared Memory Dump File	20-25
onstat -a command: Print overall status of the database server	20-26
onstat -b command: Print buffer information for buffers in use	20-26
onstat -B command: Prints information about used buffers	20-27
onstat -c command: Print ONCONFIG file contents	20-29
onstat -C command: Print B-tree scanner information	20-29
onstat -d command: Print chunk information	20-34
onstat -D command: Print page-read and page-write information	20-39
onstat -f command: Print dbspace information affected by dataskip	20-40
onstat -F command: Print counts	20-40
onstat -g monitoring options	20-42
onstat -g act command: Print active threads	20-48
onstat -g afr command: Print allocated memory fragments	20-48
onstat -g all command: Print diagnostic information.	20-49
onstat -g aqt command: Print data mart and accelerated query table information	20-49
onstat -g arc command: Print archive status	20-52
onstat -g ath command: Print information about all threads	20-54
onstat -g buf command: Print buffer pool profile information	20-55
onstat -g cat command: Print ER global catalog information	20-57
onstat -g cdr command: Print ER statistics	20-58
onstat -g cdr config command: Print ER settings	20-59
onstat -g ckp command: Print checkpoint history and configuration recommendations	20-61
onstat -g cluster command: Print high-availability cluster information.	20-65
onstat -g cmsm command: Print Connection Manager information	20-69
onstat -g con command: Print condition and thread information.	20-71
onstat -g cpu : Print runtime statistics.	20-71
onstat -g dbc command: Print dbScheduler and dbWorker thread statistics	20-73
onstat -g ddr command: Print ER database log reader status	20-74
onstat -g defragment command: Print defragment partition extents	20-75
onstat -g dic command: Print table information	20-76
onstat -g dis command: Print database server information.	20-77
onstat -g dsk command: Print the progress of the currently running compression operation	20-78
onstat -g dll command: Print dynamic link library file list.	20-79
onstat -g dmp command: Print raw memory	20-80
onstat -g dri command: Print high-availability data replication information	20-81
onstat -g dsc command: Print distribution cache information	20-83
onstat -g dss command: Print ER environment data	20-84
onstat -g dtc command: Print delete table cleaner statistics	20-85
onstat -g env command: Print environment variable values	20-86
onstat -g ffr command: Print free fragments	20-88
onstat -g glo command: Print global multithreading information	20-89
onstat -g grp command: Print ER grouper statistics	20-91
onstat -g his command: Print SQL trace information.	20-97
onstat -g ioa command: Print combined onstat -g information	20-102
onstat -g iob command: Print big buffer use summary	20-103
onstat -g iof command: Print asynchronous I/O statistics	20-104
onstat -g iog command: Print AIO global information.	20-105
onstat -g ioq command: Print I/O queue information	20-105
onstat -g ipl command: Print index page logging status information	20-107
onstat -g iov command: Print AIO VP statistics	20-107
onstat -g lap command: Print light appends status information.	20-109
onstat -g laq command: Print secondary server queues	20-109
onstat -g lmm command: Print low memory management information	20-111
onstat -g lmx command: Print all locked mutexes	20-113
onstat -g lsc command: Print active light scan status (deprecated)	20-114
onstat -g mem command: Print pool memory statistics	20-114

onstat -g mgm command: Print MGM resource information	20-116
onstat -g nbm command: Print a block bit map	20-119
onstat -g nif command: Print statistics about the network interface	20-120
onstat -g nsc command: Print current shared memory connection information.	20-121
onstat -g nsd command: Print poll threads shared-memory data	20-124
onstat -g nss command: Print shared memory network connections status	20-124
onstat -g ntd command: Print network statistics	20-125
onstat -g ntm command: Print network mail statistics	20-126
onstat -g ntt command: Print network user times	20-126
onstat -g ntu command: Print network user statistics	20-127
onstat -g opn command: Print open partitions	20-127
onstat -g osi command: Print operating system information	20-129
onstat -g pos command: Print file values	20-130
onstat -g ppd command: Print partition compression dictionary information	20-131
onstat -g ppf command: Print partition profiles	20-132
onstat -g pqs command: Print operators for all SQL queries	20-133
onstat -g prc command: Print sessions using UDR or SPL routine	20-134
onstat -g proxy command: Print proxy distributor information	20-135
onstat -g que command: Prints ER queue statistics	20-140
onstat -g qst command: Print wait options for mutex and condition queues	20-141
onstat -g rah command: Print read-ahead request statistics	20-142
onstat -g rbm command: Print a block map of shared memory	20-143
onstat -g rcv command: Print ER receive manager statistics	20-144
onstat -g rea command: Print ready threads	20-147
onstat -g rep command: Print ER schedule manager events	20-148
onstat -g rqm command: Print low-level queue statistics	20-148
onstat -g rss command: Print RS secondary server information	20-151
onstat -g rwm command: Print read and write mutexes	20-154
onstat -g sch command: Print VP information	20-155
onstat -g scn command: Print scan information	20-155
onstat -g sds command: Print SD secondary server information	20-157
onstat -g seg command: Print shared memory segment statistics	20-161
onstat -g ses command: Print session-related information	20-162
onstat -g sle command: Print all sleeping threads	20-167
onstat -g smb command: Print sbspaces information	20-167
onstat -g smx command: Print multiplexer group information	20-169
onstat -g spi command: Print spin locks with long spins	20-170
onstat -g sql command: Print SQL-related session information	20-172
onstat -g src command: Patterns in shared memory	20-173
onstat -g ssc command: Print SQL statement occurrences.	20-174
onstat -g stk command: Print thread stack.	20-176
onstat -g stm command: Print SQL statement memory usage	20-176
onstat -g stq command: Print queue information	20-177
onstat -g sts command: Print stack usage for each thread.	20-177
onstat -g sym command: Print symbol table information for the oninit utility.	20-178
onstat -g sync command: Print ER synchronization status	20-179
onstat -g tpf command: Print thread profiles	20-180
onstat -g ufr command: Print memory pool fragments	20-181
onstat -g vpcache command: Print CPU VP memory block cache statistics	20-182
onstat -g wai command: Print wait queue thread list	20-184
onstat -g wmx command: Print all mutexes with waiters	20-184
onstat -g wst command: Print wait statistics for threads	20-185
onstat -G command: Print TP/XA transaction information	20-187
onstat -h command: Print buffer header hash chain information	20-188
onstat -i command: Initiate interactive mode	20-189
onstat -j command: Provide onpload status information	20-190
onstat -k command: Print active lock information	20-192
onstat -l command: Print physical and logical log information	20-193
onstat -L command: Print the number of free locks.	20-196
onstat -m command: Print recent system message log information.	20-197
onstat -o command: Output shared memory contents to a file	20-198

onstat -O command: Print optical subsystem information	20-198
onstat -p command: Print profile counts	20-200
onstat -P command: Print partition information	20-203
onstat -r command: Repeatedly print selected statistics	20-204
onstat -R command: Print LRU, FLRU, and MLRU queue information	20-207
onstat -s command: Print latch information	20-209
onstat -t and onstat -T commands: Print tblspace information	20-211
onstat -u command: Print user activity profile	20-212
onstat -x command: Print database server transaction information.	20-215
Determine the position of a logical-log record	20-217
Determine the mode of a global transaction	20-217
onstat -X command: Print thread information	20-217
onstat -z command: Clear statistics	20-219
Return codes on exiting the onstat utility	20-219

Part 3. SQL Administration API

Chapter 21. SQL Administration API Functions. 21-1

SQL Administration API Overview	21-1
admin() and task() Function Syntax Behavior	21-1
admin() and task() Argument Size Specifications	21-2
admin() and task() Function Return Codes	21-2
SQL administration API portal: Arguments by functional category	21-4
add bufferpool argument: Add a buffer pool (SQL administration API)	21-14
add chunk argument: Add a new chunk (SQL administration API)	21-15
add log argument: Add a new logical log (SQL administration API)	21-16
add memory argument: Increase shared memory (SQL administration API)	21-17
add mirror argument: Add a mirror chunk (SQL administration API)	21-18
alter chunk argument: Change chunk status to online or offline (SQL administration API)	21-19
alter logmode argument: Change the database logging mode (SQL administration API)	21-20
alter plog argument: Change the physical log (SQL administration API)	21-21
archive fake argument: Perform an unrecorded backup (SQL administration API)	21-21
cdr argument: Administer Enterprise Replication (SQL administration API)	21-22
check data argument: Check data consistency (SQL administration API)	21-23
check extents argument: Check extent consistency (SQL administration API).	21-24
check partition argument: Check partition consistency (SQL administration API)	21-24
checkpoint argument: Force a checkpoint (SQL administration API)	21-25
clean sbospace argument: Release unreferenced smart large objects (SQL administration API)	21-26
create blobospace argument: Create a blobospace (SQL administration API).	21-26
create blobospace from storagepool argument: Create a blobospace from the storage pool (SQL administration API).	21-28
create chunk argument: Create a chunk (SQL administration API)	21-29
create chunk from storagepool argument: Create a chunk from the storage pool (SQL administration API).	21-30
create database argument: Create a database (SQL administration API)	21-31
create dbspace argument: Create a dbspace (SQL administration API)	21-32
create dbspace from storagepool argument: Create a dbspace from the storage pool (SQL administration API)	21-33
create sbospace argument: Create an sbospace (SQL administration API).	21-34
create sbospace from storagepool argument: Create an sbospace from the storage pool (SQL administration API)	21-35
create sbospace with accesstime argument: Create an sbospace that tracks access time (SQL administration API)	21-36
create sbospace with log argument: Create an sbospace with transaction logging (SQL administration API)	21-37
create tempdbspace argument: Create a temporary dbspace (SQL administration API)	21-38
create tempdbspace from storagepool argument: Create a temporary dbspace from the storage pool (SQL administration API).	21-39
create tempsbospace argument: Create a temporary sbospace (SQL administration API).	21-39
create tempsbospace from storagepool argument: Create a temporary sbospace from the storage pool (SQL administration API).	21-41
defragment argument: Dynamically defragment partition extents	21-41
drop blobospace argument: Drop a blobospace (SQL administration API)	21-43
drop blobospace to storagepool argument: Return space from an empty blobospace to the storage pool (SQL administration API).	21-44

drop chunk argument: Drop a chunk (SQL administration API)	21-44
drop chunk to storagepool argument: Return space from an empty chunk to the storage pool (SQL administration API).	21-45
drop database argument: Drop a database (SQL administration API)	21-46
drop dbspace argument: Drop a dbspace (SQL administration API).	21-46
drop dbspace to storagepool argument: Return space from an empty dbspace to the storage pool (SQL administration API).	21-47
drop log argument: Drop a logical log (SQL administration API)	21-47
drop sbspace argument: Drop an sbspace (SQL administration API)	21-49
drop sbspace to storagepool argument: Return space from an empty sbspace to the storage pool (SQL administration API).	21-49
drop tempdbspace argument: Drop a temporary dbspace (SQL administration API)	21-50
drop tempdbspace to storagepool argument: Return space from an empty temporary dbspace to the storage pool (SQL administration API)	21-50
drop tempsbspace to storagepool argument: Return space from an empty temporary sbspace to the storage pool (SQL administration API)	21-51
file status argument: Display the status of a message log file (SQL administration API)	21-51
ha make primary argument: Change the mode of a secondary server (SQL administration API)	21-52
ha rss argument: Create an RS secondary server (SQL administration API)	21-53
ha rss add argument: Add an RS secondary server to a primary server (SQL administration API)	21-54
ha rss change argument: Change the password of an RS secondary server (SQL administration API)	21-55
ha rss delete argument: Delete an RS secondary server (SQL administration API)	21-56
ha sds clear argument: Stop shared-disk replication (SQL administration API)	21-56
ha sds primary argument: Convert an SD secondary server to a primary server (SQL administration API).	21-57
ha sds set argument: Create a shared-disk primary server (SQL administration API)	21-58
ha set idxauto argument: Replicate indexes to secondary servers (SQL administration API).	21-59
ha set ipl argument: Log index builds on the primary server (SQL administration API)	21-59
ha set primary argument: Define an HDR primary server (SQL administration API)	21-60
ha set secondary argument: Define an HDR secondary server (SQL administration API).	21-61
ha set standard argument: Convert an HDR server into a standard server (SQL administration API).	21-61
ha set timeout argument: Change SD secondary server timeout (SQL administration API)	21-62
message log delete argument: Delete a message log file (SQL administration API)	21-63
message log rotate argument: Rotate the message log file (SQL administration API)	21-63
message log truncate argument: Delete the contents of a message log file (SQL administration API)	21-64
modify chunk extend argument: Extend the size of a chunk (SQL administration API)	21-65
modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)	21-66
modify chunk extendable off argument: Mark a chunk as not extendable (SQL administration API)	21-67
modify space expand argument: Expand the size of a space (SQL administration API)	21-68
modify space sp_sizes argument: Modify the create or extend size of a storage space (SQL administration API)	21-69
onmode and a arguments: Add a shared-memory segment (SQL administration API).	21-71
onmode and c arguments: Force a checkpoint (SQL administration API)	21-72
onmode and C arguments: Control the B-tree scanner (SQL administration API)	21-73
onmode and d arguments: Set data-replication types (SQL administration API).	21-74
onmode and D arguments: Set PDQ priority (SQL administration API)	21-76
onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)	21-76
onmode and F arguments: Free unused memory segments (SQL administration API).	21-77
onmode and j arguments: Switch the database server to administration mode (SQL administration API)	21-78
onmode and l arguments: Switch to the next logical log (SQL administration API).	21-79
onmode and m arguments: Switch to multi-user mode (SQL administration API)	21-79
onmode and M arguments: Temporarily change decision-support memory (SQL administration API)	21-80
onmode and n arguments: Unlock resident memory (SQL administration API)	21-81
onmode and O arguments: Mark a disabled dbspace as down (SQL administration API)	21-81
onmode and p arguments: Add or remove virtual processors (SQL administration API)	21-82
onmode and Q arguments: Set maximum number for decision-support queries (SQL administration API)	21-83
onmode and r arguments: Force residency of shared memory (SQL administration API).	21-84
onmode and S arguments: Set maximum number of decision-support scans (SQL administration API)	21-85
onmode and W arguments: Reset statement cache attributes (SQL administration API)	21-85
onmode and wf arguments: Permanently update a configuration parameter (SQL administration API)	21-87
onmode and wm arguments: Temporarily update a configuration parameter (SQL administration API).	21-88
onmode, wm, and AUTO_LRU_TUNING arguments: Change LRU tuning status (SQL administration API)	21-89
onmode and Y arguments: Change query plan measurements for a session (SQL administration API)	21-90

onmode and z arguments: Terminate a user session (SQL administration API)	21-91
onmode and Z arguments: Terminate a distributed transaction (SQL administration API)	21-91
onbar argument: Backup the storage spaces (SQL administration API)	21-92
onsmsync argument: Synchronize with the storage manager catalog (SQL administration API)	21-93
ontape archive argument: Backup the data on your database (SQL administration API)	21-95
print error argument: Print an error message (SQL administration API)	21-96
print file info argument: Display directory or file information (SQL administration API)	21-96
print partition argument: Print partition information (SQL administration API)	21-97
rename space argument: Rename a storage space (SQL administration API)	21-98
reset sysadmin argument: Move the sysadmin database (SQL administration API)	21-99
restart listen argument: Stop and start a listen thread dynamically (SQL administration API)	21-100
scheduler argument: Stop or start the scheduler (SQL administration API)	21-101
scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)	21-101
scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)	21-104
set chunk argument: Change the status of a chunk (SQL administration API)	21-105
set dataskip argument: Start or stop skipping a dbspace (SQL administration API)	21-106
set index compression argument: Change index page compression (SQL administration API)	21-106
set onconfig memory argument: Temporarily change a configuration parameter (SQL administration API)	21-107
set onconfig permanent argument: Permanently change a configuration parameter (SQL administration API)	21-108
set sbospace accesstime argument: Control access time tracking (SQL administration API)	21-109
set sbospace avg_lo_size argument: Set the average size of smart large objects (SQL administration API)	21-110
set sbospace logging argument: Change the logging of an sbospace (SQL administration API)	21-111
set sql tracing argument: Set global SQL tracing (SQL administration API)	21-111
set sql tracing database argument: Change database tracing (SQL administration API)	21-113
set sql tracing session argument: Control tracing for a session (SQL administration API)	21-114
set sql tracing user argument: Control tracing for users (SQL administration API)	21-115
set sql user tracing argument: Set global SQL tracing for a user session (SQL administration API)	21-115
start listen argument: Start a listen thread dynamically (SQL administration API)	21-116
start mirroring argument: Starts storage space mirroring (SQL administration API)	21-117
stop listen argument: Stop a listen thread dynamically (SQL administration API)	21-117
stop mirroring argument: Stops storage space mirroring (SQL administration API)	21-118
storagepool add argument: Add a storage pool entry (SQL administration API)	21-119
storagepool delete argument: Delete one storage pool entry (SQL administration API)	21-121
storagepool modify argument: Modify a storage pool entry (SQL administration API)	21-122
storagepool purge argument: Delete storage pool entries (SQL administration API)	21-124
Compress and Uncompress Operations (SQL administration API)	21-125
table or fragment arguments: Compress data and optimize storage (SQL administration API)	21-126
Output of the Estimate Compression Operation (SQL administration API)	21-131
purge compression dictionary arguments: Remove compression dictionaries (SQL administration API)	21-132

Part 4. Appendixes

Appendix A. Files That the Database Server Uses. A-1

Database Server Files.	A-1
Descriptions of Files	A-4
af.xxx	A-4
ac_msg.log	A-4
ac_config.std	A-4
bar_act.log	A-4
bldutil.process_id	A-5
buildsmi.out (UNIX) or buildsmi_out (Windows)	A-5
concdr.sh.	A-5
.conf.dbservername	A-5
core	A-5
Emergency Boot Files for ON-Bar	A-5
gcore.xxx (UNIX)	A-6
illlsrra.xx	A-6
~/informix	A-6
informix.rc (UNIX)	A-6

INFORMIXTMP	A-6
.inf.servicename	A-7
.infos.dbservername	A-7
.infxdirs	A-7
InstallServer.log (Windows)	A-7
ISM Catalog.	A-7
ISM Logs.	A-7
ISMversion	A-8
JVM_vpid file	A-8
JVPLOG	A-8
.jvpprops.	A-8
Message Log	A-8
onconfig.std.	A-8
The ONCONFIG File.	A-9
onconfig	A-9
oncfg_servername.servernum	A-9
onsnmp.servername.	A-10
onsrvapd.log	A-10
revcdr.sh	A-10
shmem.xxx (UNIX)	A-10
sm_versions.std	A-10
snmpd.log	A-10
sqlhosts	A-10
VP.servername.nnx	A-11
xbsa.messages.	A-11

Appendix B. Troubleshooting errors B-1

Collecting Diagnostics using onmode -I.	B-1
Creating Tracepoints	B-1
Collecting data with the ifxcollect tool	B-1

Appendix C. Event Alarms C-1

Using ALARMPROGRAM to Capture Events.	C-1
Setting ALRM_ALL_EVENTS	C-1
Writing Your Own Alarm Script	C-1
Customizing the ALARMPROGRAM Scripts	C-1
Precautions for Foreground Operations in Alarm Scripts	C-2
Interpreting event alarm messages	C-2
Events in the ph_alert Table	C-3
Event Alarm Parameters.	C-4
Event alarm IDs	C-5
Severity 5 event alarms.	C-50
RS Secondary Server Event Alarms.	C-55
SD Secondary Server Event Alarms	C-56

Appendix D. Discontinued configuration parameters D-1

AFF_NPROCS (Discontinued).	D-2
AFF_SPROC (Discontinued)	D-2
BUFFERS (Discontinued)	D-3
FAST_RESTART_CKPT_FUZZYLOG (Discontinued)	D-4
FAST_RESTART_PHYSLOG (Discontinued)	D-4
JDKVERSION (Discontinued)	D-5
LBU_PRESERVE (Discontinued)	D-5
LOGSMAX (Discontinued)	D-5
LRU_MAX_DIRTY (Discontinued)	D-6
LRU_MIN_DIRTY (Discontinued)	D-6
LRUS (Discontinued).	D-7
NOAGE (Discontinued).	D-7
NUMAIOVPS (Discontinued)	D-8
NUMCPUVPS (Discontinued)	D-8

PHYSDBS (Discontinued)	D-9
RA_THRESHOLD configuration parameter (discontinued)	D-10
SB_CHECK_FOR_TEMP configuration parameter (discontinued)	D-10

Appendix E. Error Messages E-1

How the Messages Are Ordered in This Chapter	E-1
How to View These Messages	E-1
Message Categories	E-2
Messages: A-B	E-2
Aborting Long Transaction: <i>tx 0xn</i>	E-2
Affinitied VP <i>mm</i> to phys proc <i>nn</i>	E-2
Affinity not enabled for this server.	E-3
Assert Failed: Error from SBSpace cleanup thread.	E-3
Assert Failed: Short description of what failed Who: Description of user/session/thread running at the time Result: State of the affected database server entity Action: What action the database administrator should take	
See Also: DUMPDIR/af.uniqid containing more diagnostics.. . . .	E-3
Begin re-creating indexes deferred during recovery.. . . .	E-3
Building 'sysmaster' database requires ~ <i>mm</i> pages of logical log. Currently there are <i>nn</i> pages available.	
Prepare to back up your logs soon.	E-4
Building 'sysmaster' database...	E-4
Messages: C	E-4
Cannot Allocate Physical-log File, <i>mm</i> wanted, <i>nn</i> available.. . . .	E-4
Cannot alter a table which has associated violations table.	E-4
Cannot change to mode.. . . .	E-4
Cannot Commit Partially Complete Transactions.	E-5
Cannot create a user-defined VP class with 'SINGLE_CPU_VP' non-zero.	E-5
Cannot create violations/diagnostics table.	E-5
Cannot insert from the violations table to the target table.	E-6
Cannot modify/drop a violations/diagnostics table.	E-6
Cannot Open Dbspace <i>mmn</i>	E-6
Cannot Open Logical Log.	E-7
Cannot Open Mirror Chunk <i>pathname</i> , errno = <i>nn</i>	E-7
Cannot Open Primary Chunk <i>pathname</i> , errno = <i>mmn</i>	E-7
Cannot Open Primary Chunk <i>chunkname</i>	E-7
Cannot open sysams in database <i>name</i> , iserrno <i>number</i>	E-7
Cannot open sysdistrib in database <i>name</i> , iserrno <i>number</i>	E-8
Cannot open <i>system_table</i> in database <i>name</i> , iserrno <i>number</i>	E-8
Cannot open systrigbody in database <i>name</i> , iserrno <i>number</i>	E-8
Cannot open systriggers in database <i>name</i> , iserrno <i>number</i>	E-8
Cannot open sysxtotypes in database <i>name</i> , iserrno <i>number</i>	E-8
Cannot Perform Checkpoint, shut system down.. . . .	E-8
Cannot Restore to Checkpoint.. . . .	E-9
Cannot Rollback Incomplete Transactions.	E-9
Cannot update pagezero.	E-9
Cannot update syscasts in database <i>name</i> . Iserrno <i>number</i>	E-9
Can't affinity VP <i>mm</i> to phys proc <i>nn</i>	E-10
Changing the sbpace minimum extent value: old value <i>value1</i> , new value <i>value2</i>	E-10
Checkpoint blocked by down space, waiting for override or shutdown.	E-10
Checkpoint Completed: duration was <i>n</i> seconds.	E-10
Checkpoint Page Write Error.. . . .	E-10
Checkpoint Record Not Found in Logical Log.	E-11
Chunk <i>chunkname</i> added to space <i>spacename</i>	E-11
Chunk <i>chunkname</i> dropped from space <i>spacename</i>	E-11
Chunk <i>number nn pathname</i> -- Offline.	E-11
Chunk <i>number nn pathname</i> -- Online.	E-11
The chunk <i>pathname</i> must have READ/WRITE permissions for owner and group.	E-12
The chunk <i>pathname</i> must have <i>owner-ID</i> and <i>group-ID</i> set to informix.	E-12
The chunk <i>pathname</i> will not fit in the space specified.	E-12
Cleaning stray LOs in sbpace <i>sbspacename</i>	E-12
Completed re-creating indexes.	E-12
Configuration has been grown to handle up to <i>integer</i> chunks.	E-13

Configuration has been grown to handle up to <i>integer</i> dbslices.	E-13
Configuration has been grown to handle up to <i>integer</i> dbspaces.	E-13
Continuing Long Transaction (for COMMIT): <i>tx 0xn</i>	E-13
Could not disable priority aging: <i>errno = number</i>	E-13
Could not fork a virtual processor: <i>errno = number</i>	E-14
Create_vp: cannot allocate memory.	E-14
Messages: D-E-F	E-14
Dataskip is OFF for all dbspaces.	E-14
Dataskip is ON for all dbspaces.. . . .	E-14
Dataskip is ON for dbspaces: <i>dbspacelist</i>	E-14
Dataskip will be turned {ON OFF} for <i>dbspacename</i>	E-15
DBSERVERALIAS exceeded the maximum limit of 32.	E-15
DBSPACETEMP internal list not initialized, using default.	E-15
The DBspace/BLOBspace <i>spacename</i> is now mirrored.	E-15
The DBspace/BLOBspace <i>spacename</i> is no longer mirrored.. . . .	E-15
Dbpace <i>dbspacename</i> for Physical-log File not found.	E-16
<i>devname</i> : write failed, file system is full.	E-16
Dropping temporary tblspace <i>0xn</i> , recovering <i>nn</i> pages.	E-16
Dynamically allocated new shared memory segment (size <i>nnnn</i>).. . . .	E-16
ERROR: NO "wait for" locks in Critical Section.	E-16
Error building sysmaster database. See <i>outfile</i>	E-17
Error in dropping system defined type.	E-17
Error in renaming systdist.	E-17
Error removing sysdistrib row for <i>tabid = tabid</i> , <i>colid = colid</i> in database <i>name</i> . <i>iserrno = number</i>	E-17
Error writing <i>pathname</i> <i>errno = number</i>	E-17
Error writing shmenv to file <i>filename</i> (<i>error</i>). Unable to create output file <i>filename</i> <i>errno=mm</i> .Error writing <i>filename</i> <i>errno=nn</i>	E-18
Fail to extend physical log space.	E-18
Fatal error initializing CWD string. Check permissions on current working directory. Group <i>groupname</i> must have at least execute permission on ' <i>.</i> '.	E-18
The following tables have outstanding old version data pages due to an In-Place Alter Table. Perform UPDATE <i>tablename</i> SET <i>column = column</i> WHERE 1=1; to clear these pages from the following tables.	E-18
Fragments <i>dbspacename1 dbspacename2</i> of table <i>tablename</i> set to non-resident.	E-19
Forced-resident shared memory not available.	E-19
Freed <i>nn</i> shared-memory segment(s) <i>number</i> bytes.	E-19
Messages: G-H-I	E-19
gcore <i>pid</i> ; mv <i>core.pid</i> dir/ <i>core.pid</i> .ABORT.	E-19
I/O <i>function</i> chunk <i>mm</i> , <i>pagenum nn</i> , <i>pagecnt aa</i> --> <i>errno = bb</i>	E-20
I/O error, <i>primary/mirror</i> Chunk <i>pathname</i> -- Offline (<i>sanity</i>).	E-20
Informix <i>database_server</i> Initialized - Complete Disk Initialized.	E-20
Informix <i>database_server</i> Initialized - Shared Memory Initialized.	E-20
Informix <i>database_server</i> Stopped.	E-21
ERROR: Insufficient available disk in the root dbspace to increase the entire Configuration save area.	E-21
Insufficient available disk in the root dbspace for the CM save area. Increase the size of the root dbspace in the ONCONFIG file and reinitialize the server.	E-21
Internal overflow of <i>shmids</i> , increase system max shared memory segment size.	E-22
Messages: J-K-L-M	E-22
Listener-thread <i>err = error_number: error_message</i>	E-22
Lock table overflow - user id <i>mm</i> session id <i>nn</i>	E-22
Logical-log File not found..	E-22
Logical Log <i>nn</i> Complete.	E-22
Logical logging <i>verror</i> for <i>type:subtype</i> in (<i>failed_system</i>).. . . .	E-23
Log Record: log = <i>ll</i> , pos = <i>0xn</i> , type = <i>type:subtype(snum)</i> , trans = <i>xx</i>	E-23
Log record (<i>type:subtype</i>) at log <i>nn</i> , <i>0xn</i> was not undone..	E-23
Log record (<i>type:subtype</i>) failed, partnum <i>pnum</i> row <i>rid</i> <i>iserrno num</i>	E-24
Log record (<i>type:subtype</i>) in log <i>nn</i> , offset <i>0xn</i> was not rolled back.	E-24
Logical Recovery allocating <i>nn</i> worker threads <i>thread_type</i>	E-24
Logical Recovery Started.	E-25
Maximum server connections <i>number</i>	E-25
Memory allocation error.	E-25
Mirror Chunk <i>chunkname</i> added to space <i>spacename</i> . Perform manual recovery.	E-25

Mixed transaction result. (<i>pid=nn user=userid</i>).	E-26
mt_shm_free_pool: pool <i>0xn</i> has blocks still used (id <i>nn</i>).	E-26
mt_shm_init: can't create <i>resident/virtual</i> segment.	E-26
mt_shm_remove: WARNING: may not have removed all/correct segments.	E-26
Messages: N-O-P.	E-27
Newly specified value of <i>value</i> for the pagesize in the configuration file does not match older value of <i>value</i> .	
Using the older value.	E-27
Not enough main memory.	E-27
Not enough logical-log files, Increase LOGFILES.	E-27
Not enough physical procs for affinity.	E-27
The number of configured CPU poll threads exceeds NUMCPUVPS.	E-28
onconfig parameter <i>parameter</i> modified from <i>old_value</i> to <i>new_value</i> .	E-28
oninit: Cannot have SINGLE_CPU_VP non-zero and number of CPU VPs greater than 1.	E-28
oninit: Cannot have SINGLE_CPU_VP non-zero and user-defined VP classes.	E-28
oninit: Cannot mix VPCLASS cpu and NUMCPUVPS, AFF_SPROC, AFF_NPROCS, or NOAGE parameters.	E-29
oninit: Cannot mix VPCLASS aio and NUMAIOVPS parameters.	E-29
oninit: Fatal error in initializing ASF with 'ASF_INIT_DATA' flags asfcode = '25507'.	E-29
oninit: invalid or missing name for Subsystem Staging Blobspace.	E-29
Cannot alter a table which has associated violations table.	E-30
oninit: Too many VPCLASS parameters specified.	E-30
oninit: VPCLASS <i>classname</i> bad affinity specification.	E-30
oninit: VPCLASS <i>classname</i> duplicate class <i>name</i> .	E-30
oninit: VPCLASS <i>classname</i> illegal option.	E-30
oninit: VPCLASS <i>classname</i> maximum number of VPs is out of the range 0-10000.	E-31
oninit: VPCLASS <i>classname</i> name is too long. Maximum length is <i>maxlength</i> .	E-31
oninit: VPCLASS <i>classname</i> number of VPs is greater than the maximum specified.	E-31
oninit: VPCLASS <i>classname</i> number of VPs is out of the range 0-10000.	E-31
onmode: VPCLASS <i>classname</i> name is too long. Maximum length is <i>maxlength</i> .	E-32
Optical Subsystem is running.	E-32
Optical Subsystem is not running.	E-32
Optical Subsystem STARTUP Error.	E-32
Online Mode.	E-32
onspaces: unable to reset dataskip.	E-33
Open transaction detected when changing log versions.	E-33
Out of message shared memory.	E-33
Out of resident shared memory.	E-33
Out of virtual shared memory.	E-33
PANIC: Attempting to bring system down.	E-34
Participant site <i>database_server</i> heuristically rolled back.	E-34
Physical recovery complete: <i>number</i> pages examined, <i>number</i> pages restored.	E-34
Physical recovery started at page (<i>chunk:offset</i>).	E-34
Portions of partition partnum of table tablename in database dbname were not logged. This partition cannot be rolled forward.	E-35
Possible mixed transaction result.	E-35
Prepared participant site <i>server_name</i> did not respond.	E-35
Prepared participant site <i>server_name</i> not responding.	E-35
Messages: Q-R-S.	E-36
Quiescent Mode.	E-36
Read failed. Table <i>name</i> , Database <i>name</i> , iserrno = <i>number</i>	E-36
Recovery Mode.	E-36
Recreating index: ' <i>dbname:"owner".tablename-idxname</i> '.	E-36
Rollforward of log record failed, iserrno = <i>nn</i> .	E-37
Root chunk is full and no additional pages could be allocated to chunk descriptor page.	E-37
scan_logundo: subsys <i>ss</i> , type <i>tt</i> , iserrno <i>ee</i> .	E-37
Session completed abnormally. Committing tx id <i>0xm</i> , flags <i>0xn</i> .	E-37
Session completed abnormally. Rolling back tx id <i>0xm</i> , flags <i>0xn</i> .	E-38
semctl: errno = <i>nn</i> .	E-38
semget: errno = <i>nn</i> .	E-38
shmat: <i>some_string</i> os_errno: <i>os_err_text</i> .	E-38
shmctl: errno = <i>nn</i> .	E-38
shmdt: errno = <i>nn</i> .	E-39

shmem sent to <i>filename</i> ..	E-39
shmget: <i>some_str</i> os_errno: key <i>shmkey</i> : <i>some_string</i> ..	E-39
Shutdown (onmode -k) or override (onmode -O)..	E-39
Shutdown Mode..	E-40
Space <i>spacename</i> added.	E-40
Space <i>spacename</i> dropped.	E-40
Space <i>spacename</i> -- Recovery Begins(<i>addr</i>)..	E-40
Space <i>spacename</i> -- Recovery Complete(<i>addr</i>)..	E-40
Space <i>spacename</i> -- Recovery Failed(<i>addr</i>)..	E-41
sysmaster database built successfully.	E-41
Successfully extend physical log space.	E-41
Messages: T-U-V	E-41
This ddl operation is not allowed due to deferred constraints pending on this table and dependent tables.	E-41
This type of space does not accept log files.	E-42
TIMER VP: Could not redirect I/O in initialization, errno = <i>nn</i> .	E-42
Too Many Active Transactions.	E-42
Too many violations.	E-42
Transaction Not Found..	E-43
Transaction heuristically rolled back.	E-43
Transaction table overflow - user id <i>nn</i> , process id <i>nn</i> .	E-43
Unable to create output file <i>filename</i> errno = <i>nn</i> .	E-43
Unable to extend <i>nn</i> reserved pages for <i>purpose</i> in root chunk.	E-43
Unable to initiate communications with the Optical Subsystem.	E-44
Unable to start SQL engine.	E-44
Unable to open tblspace <i>nn</i> , iserrno = <i>nn</i> .	E-44
The value of pagesize <i>pagesize</i> specified in the config file is not a valid pagesize. Use 2048, 4096 or 8192 as the value for PAGESIZE in the onconfig file and restart the server.	E-44
Violations table is not started for the target table.	E-45
Violations table reversion test completed successfully.	E-45
Violations table reversion test failed.	E-45
Violations table reversion test start..	E-45
Violations tables still exist..	E-45
Virtual processor limit exceeded.	E-46
VPCLASS <i>classname</i> name is too long. Maximum length is <i>maxlength</i> .	E-46
VPCLASS <i>classname</i> duplicate class name.	E-46
VPCLASS <i>classname</i> Not enough physical procs for affinity..	E-46
Messages: W-X-Y-Z	E-47
WARNING: aio_wait: errno = <i>nn</i> .	E-47
WARNING: Buffer pool size may cause database server to get into a locked state. Recommended minimum buffer pool size is <i>num</i> times maximum concurrent user threads..	E-47
warning: Chunk time stamps are invalid..	E-47
Warning: <i>name_old</i> is a deprecated onconfig parameter. Use <i>name_new</i> instead. See the release notes and the Informix Administrator's Reference for more information.	E-47
Warning: <i>name_old</i> is a deprecated onconfig parameter. Use <i>name_new</i> instead.	E-48
Warning: Unable to allocate requested big buffer of size <i>nn</i> .	E-48
You are turning off smart large object logging.	E-48
Messages: Symbols	E-48
HH:MM:SS Informix database server Version R.VV.PPPPP Software Serial Number RDS#YYYYYYY.	E-48
<i>argument</i> : invalid argument.	E-49
<i>function_name</i> : cannot allocate memory.	E-49
Conversion/Reversion Messages	E-49
Messages: A-C.	E-49
Messages: D-F.	E-52
Messages: I-P	E-54
Messages: R-W	E-58
Conversion and Reversion Messages for Enterprise Replication	E-61
CDR reversion test completed successfully.	E-62
CDR reversion test failed; for details look in \$INFORMIXDIR/etc/revtestcdr.out.	E-62
Enterprise Replication is not ready for conversion. The Control and TRG send queues should be empty for conversion/reversion to proceed.	E-62

Enterprise Replication is not ready for conversion. The syscdr database should NOT contain old-style group definitions for conversion to succeed.	E-63
Enterprise Replication should be in a stopped state for conversion/reversion to proceed.. . . .	E-63
Reversion of 'syscdr' failed; for details look in \$INFORMIXDIR/etc/revcdr.out.. . . .	E-63
Starting CDR reversion test...	E-63
Starting 'syscdr' conversion...	E-64
Starting 'syscdr' reversion...	E-64
'syscdr' conversion completed successfully.	E-64
'syscdr' conversion failed. For details, look in \$INFORMIXDIR/etc/concdr.out.	E-64
Syscdr should NOT contain new replicate sets for reversion to succeed.	E-65
Syscdr should not contain replicates defined with the --floatieee option for reversion to succeed.	E-65
Dynamic Log Messages.	E-65
Dynamically added log file <i>logid</i> to DBspace <i>dbspace_number</i>	E-65
Log file <i>logid</i> added to DBspace <i>dbspace_number</i>	E-65
Log file number <i>logid</i> has been dropped from DBspace <i>dbspace_number</i>	E-66
Log file <i>logid</i> has been pre-dropped.	E-66
Pre-dropped log file number <i>logid</i> has been deleted from DBspace <i>dbspace_number</i>	E-66
ALERT: Because the oldest logical log (<i>logid</i>) contains records from an open transaction (<i>transaction_address</i>), the server is attempting to dynamically add a log file. But there is no space available. Please add a DBspace or chunk. Then complete the transaction as soon as possible.	E-66
ALERT: The oldest logical log (<i>logid</i>) contains records from an open transaction (<i>transaction_address</i>). Logical logging will remain blocked until a log file is added. Add the log file with the onparams -a command, using the -i (insert) option, as in: onparams -a -d <i>dbspace</i> -s <i>size</i> -i. Then complete the transaction as soon as possible.. . . .	E-67
Log file <i>logid</i> has been pre-dropped. It will be deleted from the log list and its space can be reused once you take level-0 archives of all BLOBspaces, Smart BLOBspaces and non-temporary DBspaces.	E-67
Sbspace Metadata Messages	E-67
Allocated <i>number</i> pages to Metadata from chunk <i>number</i>	E-67
Allocated <i>number</i> pages to Userdata from chunk <i>number</i>	E-67
Freeing reserved space from chunk <i>number</i> to Metadata.. . . .	E-68
Freeing reserved space from chunk <i>number</i> to Userdata.. . . .	E-68
Truncate Table Messages	E-68
The table cannot be truncated if it has an open cursor or dirty readers.. . . .	E-68
The table cannot be truncated. It has at least one non-empty child table with referential constraints.	E-68

Appendix F. Limits in IBM Informix F-1

Limitations on UNIX Operating Systems	F-1
System-Level Parameter Limits (UNIX)	F-1
Table-Level Parameter Limits (UNIX)	F-1
Access capabilities (UNIX)	F-2
IBM Informix System Defaults (UNIX)	F-2
ON-Monitor Statistics (UNIX)	F-3
Limitations on Windows Operating Systems	F-3
System-Level Parameter Limits (Windows)	F-3
Table-Level Parameter Limits (Windows)	F-3
Access Capabilities (Windows).	F-4
IBM Informix System Defaults (Windows)	F-5

Appendix G. Accessibility G-1

Accessibility features for IBM Informix products	G-1
Accessibility features.	G-1
Keyboard navigation.	G-1
Related accessibility information	G-1
IBM and accessibility.	G-1
Dotted decimal syntax diagrams	G-1

Notices H-1

Trademarks	H-3
----------------------	-----

Index X-1

Introduction

In This Introduction

This introduction provides an overview of the information in this publication and describes the conventions that this publication uses.

About This Publication

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

This section discusses the intended audience for this publication and the associated software products that you must have to use the administrative utilities.

Types of Users

This publication is written for the following users:

- Database administrators
- System administrators
- Performance engineers

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

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

You can access the Informix information centers, as well as other technical information such as technotes, white papers, and IBM Redbooks® publications online at <http://www.ibm.com/software/data/sw-library/>.

Software Dependencies

This publication is written with the assumption that you are using IBM Informix or IBM Informix Dynamic Server with J/Foundation, Version 11.70, as your database server.

Assumptions About Your Locale

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

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

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

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

Demonstration Databases

The DB-Access utility includes one or more demonstration databases that you can use to learn and test with. After you add, delete, or change the data and scripts that are in the database, you can re-initialize the database to its original condition.

The demonstration databases are:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix publications are based on the **stores_demo** database.
- The **sales_demo** database provides an example of a simple data-warehousing environment and works in conjunction with the **stores_demo** database. The scripts for the **sales_demo** database create new tables and add extra rows to the items and orders tables of **stores_demo** database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

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

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

Related concepts:



The **stores_demo** Database (SQL Reference)

What's New in Administrator's Reference for Informix database server, Version 11.70

This publication includes information about new features and changes in existing functionality.

The following changes and enhancements are relevant to this publication. For a complete list of what's new in this release, see the release notes or the information center at http://publib.boulder.ibm.com/infocenter/idshelp/v117/topic/com.ibm.po.doc/new_features.htm..

The following table lists the new features for Version 11.70.xC4.

Table 1. What's New in IBM Informix Administrator's Reference for Version 11.70.xC4

Overview	Reference
<p>Use a file to authenticate server connections in a secured network environment</p> <p>You can use the S6_USE_REMOTE_SERVER_CFG configuration parameter with the REMOTE_SERVER_CFG configuration parameter to specify the file that is used to authenticate connections in a secured network environment.</p>	<p>"S6_USE_REMOTE_SERVER_CFG configuration parameter" on page 1-121</p>
<p>Configure secure connections for replication servers</p> <p>You can use the connection security option in the sqlhosts file and create an encrypted password file so that the Connection Manager and the CDR utility can securely connect to networked servers. Use the onpassword utility to encrypt or decrypt a password file.</p>	<p>Chapter 17, "The onpassword utility," on page 17-1</p>
<p>Data sampling for update statistics operations</p> <p>If you have a large index with more than 100 000 leaf pages, you can generate index statistics based on sampling when you run UPDATE STATISTICS statements in LOW mode. Gathering index statistics from sampled data can increase the speed of the update statistics operations. To enable sampling, set the USTLOW_SAMPLE configuration parameter or the USTLOW_SAMPLE option of the SET ENVIRONMENT statement.</p>	<p>"USTLOW_SAMPLE configuration parameter" on page 1-155</p>
<p>SQL administration API command arguments for creating sbspaces</p> <p>Three SQL administration API command arguments were added: create tempsbpace (creates temporary sbspaces), create sbspace with log (creates sbspaces that have transaction logging turned on), create sbspace with accesstime (creates sbspaces that track the time of access for all smart large objects stored in the sbspace).</p>	<p>"create tempsbpace argument: Create a temporary sbspace (SQL administration API)" on page 21-39</p> <p>"create sbspace with log argument: Create an sbspace with transaction logging (SQL administration API)" on page 21-37</p> <p>"create sbspace with accesstime argument: Create an sbspace that tracks access time (SQL administration API)" on page 21-36</p>
<p>Dynamically change additional configuration parameters</p> <p>In past releases you could dynamically change some configuration parameters with the onmode -wf or onmode -wm commands. As of this fix pack, you can use those commands to dynamically change these additional configurations parameters: ALARMPROGRAM, AUTO_REPREPARE, BLOCKTIMEOUT, CKPTINTVL, DBSPACETEMP, DEADLOCK_TIMEOUT, DEF_TABLE_LOCKMODE, DIRECTIVES, DRINTERVAL, DRTIMEOUT, FILLFACTOR, LOGSIZE, LTAPEBLK, LTAPEDEV, LTAPESIZE, MSGPATH, ONDBSPACEDOWN, OPTCOMPIND, RA_PAGES, SHMADD, STACKSIZE, SYSALARMPROGRAM, TAPEBLK, TAPEDEV, TAPESIZE, TBLTBLFIRST, TBLTBLNEXT, TXTIMEOUT, and WSTATS.</p>	<p>"onmode -wf, -wm: Dynamically change certain configuration parameters" on page 14-25</p>

Plan responses to high severity event alarms	"Event alarm IDs" on page C-5
You can plan responses to severity 4 and 5 event alarms based on the documented explanations and user actions.	"Severity 5 event alarms" on page C-50
Monitor client program database usage	" onstat -g ses command: Print session-related information" on page 20-162
You can use the onstat -g ses sessionid command to display the full path of the client program that is used in your session. Use the client program information to monitor or stop access to the database.	
Progress of compression operations	" onstat -g dsk command: Print the progress of the currently running compression operation" on page 20-78
When you run an onstat -g dsk command to display information about currently running compression operations, you can find useful information in two new columns: The Approx Prog column shows the percentage completed of the operation that is in progress, and the Approx Remaining column shows the approximate time remaining that is required to complete the operation.	
Prevent failover if the primary server is active	"SDS_LOGCHECK configuration parameter" on page 1-124
Use the SDS_LOGCHECK configuration parameter to prevent a shared-disk (SD) secondary server from taking over the role of the primary server. Failover to the SD secondary server is initiated if the primary server is inactive, and is not allowed if the primary server is active. This configuration parameter is useful if you do not have I/O fencing configured and your system consists of a primary server and one or more SD secondary servers.	
Global Security Kit (GSKit) support	"GSKIT_VERSION configuration parameter" on page 1-80
Informix 11.70 now supports GSKit version 7 or later, and ships with GSKit version 8. You can set the GSKIT_VERSION configuration parameter so that the database server uses a version other than GSKit version 8.	

The following table lists the new features for Version 11.70.xC3.

Table 2. What's New in IBM Informix Administrator's Reference for Version 11.70.xC3

Overview	Reference
Automatic read-ahead operations	"AUTO_READAHEAD configuration parameter" on page 1-31
You can enable the database server to use read-ahead operations automatically to improve performance. Most queries can benefit from processing the query while asynchronously retrieving the data required by the query. The database server can automatically use asynchronous operations for data or it can avoid them if the data for the query is already cached. Use the AUTO_READAHEAD configuration parameter to configure automatic read-ahead operations for all queries, and use the SET ENVIRONMENT AUTO_READAHEAD statement to configure automatic read-ahead operations for a particular session.	" onstat -g rah command: Print read-ahead request statistics" on page 20-142
The RA_THRESHOLD configuration parameter is deprecated with this release.	

Table 2. What's New in IBM Informix Administrator's Reference for Version 11.70.xC3 (continued)

Overview	Reference
<p>Managing message logs in embedded and enterprise environments</p> <p>You can use Scheduler tasks to reduce the size of message log files by automatically truncating or deleting the log files or by configuring automatic file rotation. Additionally, you can use the related ph_threshold table parameters to specify the maximum number of message log files to retain. These tasks and parameters are useful for embedded applications because they reduce DBA or system administrator requirements for managing the log files.</p> <p>You can also use SQL administration API commands to manage the size of the log files on demand, as necessary.</p>	<p>"message log rotate argument: Rotate the message log file (SQL administration API)" on page 21-63</p> <p>"message log truncate argument: Delete the contents of a message log file (SQL administration API)" on page 21-64</p> <p>"message log delete argument: Delete a message log file (SQL administration API)" on page 21-63</p> <p>"file status argument: Display the status of a message log file (SQL administration API)" on page 21-51</p>
<p>Configuring the server response to low memory</p> <p>You can configure the actions that the server takes to continue processing when memory is critically low. You can specify the criteria for terminating sessions based on idle time, memory usage, and other factors so that the targeted application can continue and avoid out-of-memory problems. Configuring the low memory response is useful for embedded applications that have memory limitations.</p>	<p>"LOW_MEMORY_MGR configuration parameter" on page 1-89</p> <p>"onstat -g lmm command: Print low memory management information" on page 20-111</p> <p>"scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)" on page 21-101</p> <p>"scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)" on page 21-104</p>
<p>Reserving memory for critical activities</p> <p>You can enable the server to reserve a specific amount of memory for use when critical activities (such as rollback activities) are needed and the server has limited free memory. If you enable the new LOW_MEMORY_RESERVE configuration parameter by setting it to a specified value in kilobytes, the critical activities can complete even when you get out-of-memory errors. You can also dynamically adjust the value of the LOW_MEMORY_RESERVE configuration parameter with the onmode -wm or -wf command.</p>	<p>"LOW_MEMORY_RESERVE configuration parameter" on page 1-90</p>

Table 2. What's New in IBM Informix Administrator's Reference for Version 11.70.xC3 (continued)

Overview	Reference
<p>Connection Manager enhancements</p> <p>You can configure a single Connection Manager instance to automatically manage client connection requests for a combination of high availability clusters, grids, server sets, and Enterprise Replication (ER) servers.</p> <p>Before version 3.70.xC3 you had to use a separate Connection Manager instance to manage each type of connection unit. For example, you had to use one Connection Manager instance for database servers that were in a grid and another Connection Manager instance for database servers that were in a high-availability cluster.</p> <p>To make configuration easier, most of the command-line options are deprecated and options are set using the configuration file. In addition, the format of the configuration file is more intuitive than before.</p> <p>Because a single Connection Manager instance supports multiple connection units, you should configure backup Connection Manager instances in case the primary instance fails.</p>	<p>Chapter 10, "The oncmsm utility," on page 10-1</p>

The following table lists the new features for Version 11.70.xC2.

Table 3. What's New in IBM Informix Administrator's Reference for Version 11.70.xC2

Overview	Reference
<p>Improve network connection performance and scalability</p> <p>You can improve the performance and scalability of network connections on UNIX operating systems by using the NUMFDSERVERS configuration parameter. Use the configuration parameter to adjust the maximum number of poll threads for the server to use when migrating TCP/IP connection across virtual processors (VPs). Adjusting the number of poll threads, for example, by increasing the number of poll threads beyond the default number, is useful if the database server has a high rate of new connect and disconnect requests or if you find a high amount of contention between network shared file (NSF) locks. For example, if you have multiple CPU VPs and poll threads and this results in NSF locking, you can increase the value of NUMFDSERVERS (and you can increase the number of poll threads specified in the NETTYPE configuration parameter) to reduce NSF lock contention.</p>	<p>"NUMFDSERVERS configuration parameter" on page 1-102</p>

Table 3. What's New in IBM Informix Administrator's Reference for Version 11.70.xC2 (continued)

Overview	Reference
<p>New SQL administration API arguments</p> <p>The following new arguments are available with the SQL admin API task() and admin() functions:</p> <ul style="list-style-type: none"> • create database: This argument creates a database and is equivalent to the CREATE DATABASE statement. • drop database: This argument drops a database and is equivalent to the DROP DATABASE statement. This function deletes the entire database, including all of the system catalog tables, objects, and data. • ontape archive: This argument invokes the ontape utility to create a backup. • onbar: This argument is equivalent to invoking specific options of the onbar utility to create backups. • onsmsync: This argument invokes the onsmsync utility to synchronize the sysutils database, the storage manager, and the emergency boot file. 	<p>"create database argument: Create a database (SQL administration API)" on page 21-31</p> <p>"drop database argument: Drop a database (SQL administration API)" on page 21-46</p> <p>"ontape archive argument: Backup the data on your database (SQL administration API)" on page 21-95</p> <p>"onbar argument: Backup the storage spaces (SQL administration API)" on page 21-92</p> <p>"onsmsync argument: Synchronize with the storage manager catalog (SQL administration API)" on page 21-93</p> <p>"print file info argument: Display directory or file information (SQL administration API)" on page 21-96</p>
<p>New event alarms</p> <p>Event alarm class IDs 81 and 83 were added.</p> <p>Event alarm 81 indicates logical log corruption. Event alarm 83 indicates that a shared disk secondary server could not become the primary server because the primary server was still active.</p>	<p>"Event alarm IDs" on page C-5</p>

The following table lists the new features for Version 11.70.xC1.

Table 4. What's New in IBM Informix Administrator's Reference for Version 11.70.xC1

Overview	Reference
<p>New SQLHOSTS parameter for Connection Manager</p> <p>You can set a new parameter named SQLHOSTS in the Connection Manager configuration file to specify how the Connection Manager searches for server instances that are requested by client connections. The SQLHOSTS option is useful when you want to allow user access or prevent user access to one or more servers in a high-availability cluster.</p>	<p>Chapter 10, "The oncmsm utility," on page 10-1</p>
<p>Automating application connections to Enterprise Replication servers</p> <p>You can use the Connection Manager to direct application requests to the appropriate Enterprise Replication server. If you have created tables through a grid with replication enabled, you can route client connections to Enterprise Replication servers based on the quality of replicated data and transaction latency.</p>	<p>Chapter 10, "The oncmsm utility," on page 10-1</p>

Table 4. What's New in IBM Informix Administrator's Reference for Version 11.70.xC1 (continued)

Overview	Reference
<p>Quickly clone a primary server</p> <p>You can now use the ifxclone utility to clone a primary server with minimal setup and configuration. Previously to clone a server it was necessary to create a level-0 backup, transfer the backup to the new system, restore the image, and initialize the instance. The ifxclone utility starts the backup and restore processes simultaneously and there is no need to read or write data to disk or tape. You can use the ifxclone utility to create a standalone server or a remote standalone secondary server. For example, you can quickly, easily, and securely clone a production system to a test system. The ifxclone utility requires the DIRECT_IO configuration parameter to be set to 0 on both the source and target servers.</p>	Chapter 18, "The ifxclone utility," on page 18-1
<p>Add a server to a replication domain by cloning</p> <p>You can add a replication server to an existing replication domain by using the ifxclone utility to clone an existing replication server onto a target database server.</p>	Chapter 18, "The ifxclone utility," on page 18-1
<p>Defragmenting partitions</p> <p>A frequently updated table can become fragmented over time and this fragmentation degrades performance when the table is accessed by the server. You can improve performance by defragmenting partitions to merge noncontiguous extents. Defragmenting a table brings data rows closer together and avoids partition header page overflow problems. Use the SQL administration API task() or admin() function with the defragment argument and specify the table name or partition number that you want to defragment.</p>	<p>"defragment argument: Dynamically defragment partition extents" on page 21-41</p> <p>"onstat -g defragment command: Print defragment partition extents" on page 20-75</p>
<p>Faster C user-defined routines</p> <p>You can improve the performance of C user-defined routines (UDRs) on UNIX, Linux, and Mac OS operating systems by loading the C UDR shared libraries when the database server is started. Otherwise, C UDR shared libraries are loaded when the libraries are first used. To preload a C UDR shared library, set the PRELOAD_DLL_FILE configuration parameter to the library path and file name. You must specify a separate PRELOAD_DLL_FILE configuration parameter for each library that you want to preload.</p>	<p>"PRELOAD_DLL_FILE Configuration Parameter" on page 1-112</p> <p>"onstat -g dll command: Print dynamic link library file list" on page 20-79</p>
<p>Monitoring high-availability servers</p> <p>You can now monitor the status of the primary server and all secondary servers in a high-availability cluster by using one command: onstat -g cluster. This command is an alternative to the individual commands: onstat -g dri, onstat -g sds, and onstat -g rss.</p>	" onstat -g cluster command: Print high-availability cluster information" on page 20-65
<p>Easier event alarm handling</p> <p>Event alarms now have a unique identification number for each specific message. You can write scripts to handle event alarms based on the unique identification number that corresponds to each specific message in an alarm class. Previously, event alarm handling scripts had to combine the class ID and the specific message.</p>	<p>"Event alarm IDs" on page C-5</p> <p>"Event Alarm Parameters" on page C-4</p>

Table 4. What's New in IBM Informix Administrator's Reference for Version 11.70.xC1 (continued)

Overview	Reference
<p>Fragment-level statistics</p> <p>In previous releases, for fragmented tables data distributions were calculated at table level to optimize query plans. This release supports a finer granularity of statistics for fragmented tables. The statistics are calculated and stored at the individual fragment level. Set the new STATLEVEL property of fragmented tables to specify whether TABLE or FRAGMENT is the granularity for data distributions, or set to AUTO to allow the database server to automatically choose the granularity of the distribution statistics for each fragmented table.</p>	<p>"AUTO_STAT_MODE configuration parameter" on page 1-33</p> <p>"STATCHANGE configuration parameter" on page 1-142</p>
<p>Compression functionality automatically enabled</p> <p>You can now compress data in a table or table fragment without first enabling compression. Previously you needed to run an SQL administration API command with the enable compression argument to enable the compression of data in a table or table fragment.</p>	<p>"Compress and Uncompress Operations (SQL administration API)" on page 21-125</p>

Table 4. What's New in IBM Informix Administrator's Reference for Version 11.70.xC1 (continued)

Overview	Reference
<p>Automatic storage provisioning ensures space is available</p> <p>You can configure Informix to automatically expand an existing storage space if the space is full. You can also configure Informix to expand the space before it is full, when its free pages fall below a specified threshold. When you enable and configure automatic storage provisioning, you do not need to manually add storage space to avoid out-of-space errors. Even if you prefer to add space manually, automatic storage provisioning simplifies the process of adding space, because you do not need to determine where to get the space.</p> <p>You enable or disable the automatic expansion of storage spaces by setting the new SP_AUTOEXPAND configuration parameter.</p> <p>You can expand a storage space in one of two ways:</p> <ul style="list-style-type: none"> • Enable chunk extensions by marking specific chunks as extendable, using a new SQL administrative API command. • Enable automatic chunk creation by using another new SQL administrative API command to add valid entries to the Informix storage pool. Valid entries can include available directories, cooked files, and raw devices. <p>In addition, you can use new SQL administrative API commands to:</p> <ul style="list-style-type: none"> • Manually extend a chunk. • Modify the amount by which the data server can automatically expand a particular storage space. • Disable automatic expansion for a storage space. • Manually create storage spaces and chunks from the storage pool. • Return space to the storage pool, while dropping chunks and storage spaces. • Manage the storage pool by adding, modifying, and deleting entries. <p>Automatic storage provisioning is supported in a high-availability cluster. In this environment, any storage pool entry (directory, cooked file, or raw device) on the primary server must also be available through the same path on all secondary servers.</p>	<p>"SP_AUTOEXPAND configuration parameter" on page 1-136</p> <p>"SP_THRESHOLD Configuration Parameter" on page 1-136</p> <p>"SP_WAITTIME Configuration Parameter" on page 1-137</p> <p>"Storage Provisioning Arguments" in "SQL administration API portal: Arguments by functional category" on page 21-4</p> <p>"The storagepool table" on page 3-9</p>
<p>Improving performance by reducing buffer reads</p> <p>If you enable the new BATCHEDREAD_INDEX configuration parameter, the optimizer automatically chooses to fetch a set of keys from an index buffer, reducing the number of buffer times a buffer is read.</p>	<p>"BATCHEDREAD_INDEX configuration parameter" on page 1-34</p>

Table 4. What's New in IBM Informix Administrator's Reference for Version 11.70.xC1 (continued)

Overview	Reference
<p>Improve name service connection time</p> <p>You can use the new NS_CACHE configuration parameter to define the maximum retention time for an individual entry in the host name/IP address cache, the service cache, the user cache, and the group cache. If you disable one or more of these caches by setting the retention time to 0, the database server queries the operating system for the host, service, user, or group information. Getting information from the name service cache instead of querying the operating system decreases the amount of time needed to establish connections.</p> <p>The database server also now supports multiple listener threads for one service (port) for the onsoctcp and the onimcsoc protocols.</p>	<p>"NS_CACHE configuration parameter" on page 1-101</p>
<p>Improved return codes for the oninit utility</p> <p>The oninit utility now returns specific codes for failures during server initialization instead of returning a single code regardless of the cause.</p>	<p>"Return codes for the oninit utility" on page 12-4</p>
<p>ifxcollect tool for collecting data for specific problems</p> <p>You can use the new ifxcollect tool to collect diagnostic data if necessary for troubleshooting a specific problem, such as an assertion failure. The ifxcollect tool is in the \$INFORMIXDIR/bin directory. Output files that ifxcollect commands generate are in the \$INFORMIXDIR/isa/data directory.</p>	<p>"Collecting data with the ifxcollect tool" on page B-1</p>
<p>Prevent the accidental disk initialization of your instance or another instance</p> <p>You can use the new FULL_DISK_INIT configuration parameter to prevent the major problems that can occur if you or someone else accidentally initializes your instance or another instance when the first page of the first chunk (page zero) exists at the root path location. Page zero, which is created when Informix is initialized, is the system page that contains general information about the server.</p> <p>The FULL_DISK_INIT configuration parameter specifies whether or not the disk initialization command (oninit -i) can run on your Informix instance when a page zero exists at the root path location. When this configuration parameter is set to 0, the oninit -i command runs only if there is not a page zero at the root path location.</p> <p>If you change the setting of the FULL_DISK_INIT configuration parameter to 1, the oninit -i command runs under all circumstances, but also resets the FULL_DISK_INIT configuration parameter to 0 after the disk initialization.</p>	<p>"FULL_DISK_INIT Configuration Parameter" on page 1-80</p>
<p>New onconfig parameter to specify the default escape character</p> <p>The DEFAULTESCCHAR configuration parameter specifies the default escape character that is used.</p>	<p>"DEFAULTESCCHAR configuration parameter" on page 1-53</p>

Table 4. What's New in IBM Informix Administrator's Reference for Version 11.70.xC1 (continued)

Overview	Reference
<p>New editions and product names</p> <p>IBM Informix Dynamic Server editions were withdrawn and new Informix editions are available. Some products were also renamed. The publications in the Informix library pertain to the following products:</p> <ul style="list-style-type: none"> • IBM Informix database server, formerly known as IBM Informix Dynamic Server (IDS) • IBM OpenAdmin Tool (OAT) for Informix, formerly known as OpenAdmin Tool for Informix Dynamic Server (IDS) • IBM Informix SQL Warehousing Tool, formerly known as Informix Warehouse Feature 	<p>For more information about the Informix product family, go to http://www.ibm.com/software/data/informix/.</p>

Example code conventions

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

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

```
CONNECT TO stores_demo
...

DELETE FROM customer
  WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

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

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

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

Additional documentation

Documentation about this release of IBM Informix products is available in various formats.

You can access or install the product documentation from the Quick Start CD that is shipped with Informix products. To get the most current information, see the Informix information centers at [ibm.com](http://www.ibm.com)[®]. You can access the information centers and other Informix technical information such as technotes, white papers, and IBM Redbooks publications online at <http://www.ibm.com/software/data/sw-library/>.

Compliance with industry standards

IBM Informix products are compliant with various standards.

IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

The IBM Informix Geodetic DataBlade® Module supports a subset of the data types from the *Spatial Data Transfer Standard (SDTS)*—*Federal Information Processing Standard 173*, as referenced by the document *Content Standard for Geospatial Metadata*, Federal Geographic Data Committee, June 8, 1994 (FGDC Metadata Standard).

Syntax diagrams

Syntax diagrams use special components to describe the syntax for statements and commands.

Table 5. Syntax Diagram Components







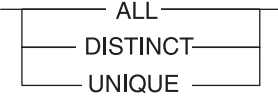
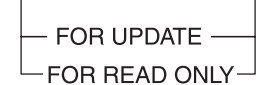
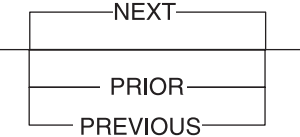
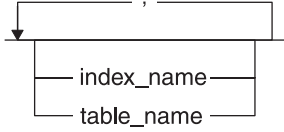
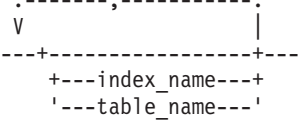

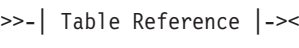
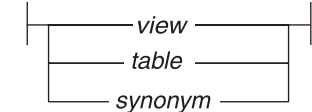
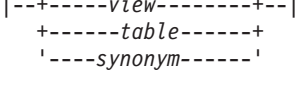
Component represented in PDF	Component represented in HTML	Meaning
	>>-----	Statement begins.
	----->	Statement continues on next line.
	>-----	Statement continues from previous line.
	-----><	Statement ends.
	-----SELECT-----	Required item.
	--+-----+-- '---LOCAL---'	Optional item.
	---+---ALL---+--- +---DISTINCT---+ '---UNIQUE---'	Required item with choice. Only one item must be present.
	---+-----+--- +---FOR UPDATE---+ '---FOR READ ONLY--'	Optional items with choice are shown below the main line, one of which you might specify.
	.---NEXT-----. ---+-----+--- +---PRIOR---+ '---PREVIOUS---'	The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line is used by default.

Table 5. Syntax Diagram Components (continued)

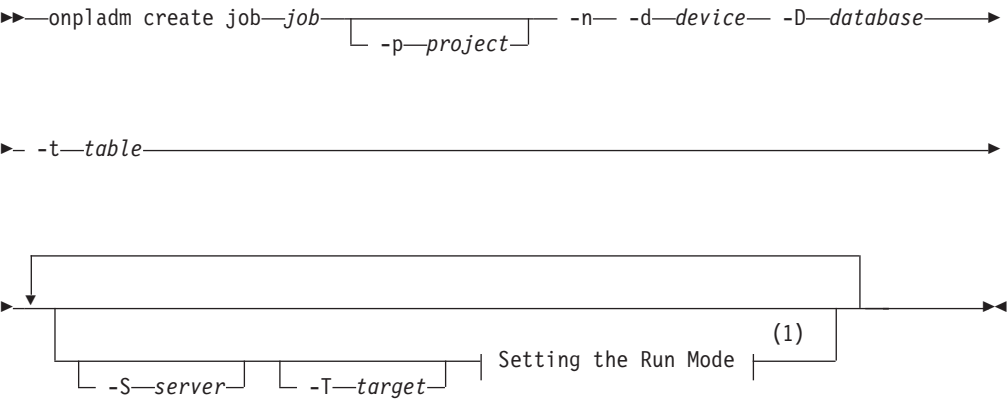
Component represented in PDF	Component represented in HTML	Meaning
		Optional items. Several items are allowed; a comma must precede each repetition.
		Reference to a syntax segment.
		Syntax segment.

How to read a command-line syntax diagram

Command-line syntax diagrams use similar elements to those of other syntax diagrams.

Some of the elements are listed in the table in Syntax Diagrams.

Creating a no-conversion job

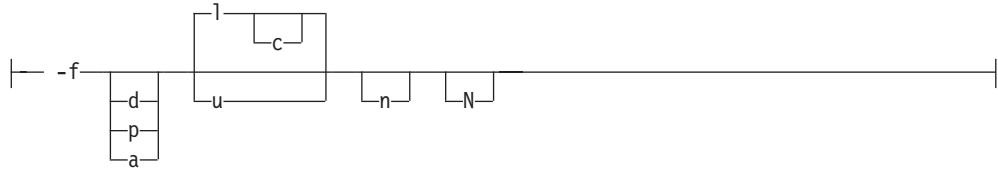


Notes:

1 See page Z-1

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

Setting the run mode:



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

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

1. Type **onpladm create job** and then the name of the job.
2. Optionally, type **-p** and then the name of the project.
3. Type the following required elements:
 - **-n**
 - **-d** and the name of the device
 - **-D** and the name of the database
 - **-t** and the name of the table
4. Optionally, you can choose one or more of the following elements and repeat them an arbitrary number of times:
 - **-S** and the server name
 - **-T** and the target server name
 - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to type **-f**, optionally type **d**, **p**, or **a**, and then optionally type **l** or **u**.
5. Follow the diagram to the terminator.

Keywords and punctuation

Keywords are words reserved for statements and all commands except system-level commands.

When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

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

Identifiers and names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples.

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

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

►►—SELECT—*column_name*—FROM—*table_name*—◄◄

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

How to provide documentation feedback

You are encouraged to send your comments about IBM Informix user documentation.

Use one of the following methods:

- Send email to docinf@us.ibm.com.
- In the Informix information center, which is available online at <http://www.ibm.com/software/data/sw-library/>, open the topic that you want to comment on. Click the feedback link at the bottom of the page, fill out the form, and submit your feedback.
- Add comments to topics directly in the information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and more!

Feedback from all methods is monitored by the team that maintains the user documentation. The feedback methods are reserved for reporting errors and omissions in the documentation. For immediate help with a technical problem, contact IBM Technical Support at <http://www.ibm.com/planetwide/>.

We appreciate your suggestions.

Part 1. Configuring and Monitoring Informix

Chapter 1. Database configuration parameters

The Informix database server uses a configuration file called ONCONFIG. The ONCONFIG file contains default configuration parameter values. You can modify the parameter values to improve performance and other characteristics of the instance or database.

onconfig File Conventions

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

In the onconfig file, each parameter is on a separate line. The file can also contain blank lines and comment lines that start with the # symbol. The maximum line limit of the onconfig file is 512 bytes. Lines that exceed this limit are truncated and might cause configuration problems.

The following line shows the syntax for a parameter line:

```
PARAMETER_NAME parameter_value comments
```

Any characters after the parameter value and blank space are interpreted as comments, regardless of whether they are preceded by a # symbol.

The parameter description and the possible values are specified in the comments above their entries in the onconfig.std file.

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

Unrecognized parameters are copied but ignored and no error is given.

Most parameters can have one valid entry. If more than one entry for these parameters exists in the onconfig file, the first entry is used. Some parameter, however, can have multiple entries, such as the DBSERVERALIASES configuration parameter.

Tip: If you use a utility like **grep** on the onconfig.std template file, specify the new line character, ^ , to return just the configuration parameter name and value. Without the new line character, the parameter description is also returned.

For example, the following command returns both the configuration parameter description and the value:

```
grep "MSGPATH" onconfig.std
# MSGPATH      - The path of the IDS message log file
MSGPATH $INFORMIXDIR/tmp/online.log
```

Whereas, the following command returns only the configuration parameter value:

```
grep "^MSGPATH" onconfig.std
MSGPATH $INFORMIXDIR/tmp/online.log
```

Creating the onconfig File

You can use the onconfig.std template file to create the configuration file that you need for the database server.

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

The IBM Informix includes the onconfig.std template file, as a template configuration file that you can copy and tailor to your specific configuration.

Do not modify or delete onconfig.std template file. This file is a template and not a functional configuration.

Important: If you omit a parameter value in your copy of the configuration file, the database server either uses default values in onconfig.std template file or calculates values based on other parameter values.


To create the onconfig file:

1. Copy the onconfig.std template file to the proper location for the onconfig file on your database server. Use the following table to determine the locations for these files.

File	Location
onconfig.std template file	UNIX: \$INFORMIXDIR/etc/onconfig.std
	Windows: %INFORMIXDIR%\etc\onconfig.std
onconfig file	UNIX: \$INFORMIXDIR/etc/\$ONCONFIG
	Windows: %INFORMIXDIR%\etc\%ONCONFIG%

2. Rename the copy that you made of the onconfig.std template file.
3. Optional. Print out a copy of the onconfig.std file to see the latest default values for the configuration parameters and recommended settings.
4. Open the onconfig file and make any changes to the configuration parameters that you need to make. For more details on why you might want to modify the default configuration parameters, see configuring the database server.
5. Set the **ONCONFIG** environment variable to the name of the onconfig file. If you do not set the environment variable, the default file name is onconfig.

Related reference:

 Initialization process (Administrator's Guide)

Displaying the Settings in the onconfig File

There are several tools that you can use to display the settings in the onconfig file.

To display the settings in the onconfig file, use one of the following tools:

- A text editor.
- The **onstat -c** command. For more information, see “**onstat -c** command: Print ONCONFIG file contents” on page 20-29.

Related reference:

 ONCONFIG environment variable (SQL Reference)

onconfig Portal: Configuration parameters by functional category

The information in this section lists configuration parameters as they appear in the `onconfig.std` file.

Category list

To use this section, you first determine the appropriate category from the following list, then follow the link to the configuration parameters for that category. The categories are listed in the same order as they appear in the `onconfig.std` file. Parameters that are not in the `onconfig.std` file but that you can add to your ONCONFIG file are listed in Table 1-53 on page 1-24.

- “Root dbspace configuration parameters” on page 1-4
- “Physical log configuration parameters” on page 1-4
- “Logical log configuration parameters” on page 1-5
- “Long transaction rollback configuration parameters” on page 1-5
- “Server message file configuration parameters” on page 1-5
- “Tblspace tblspace in the root dbspace configuration parameters” on page 1-6
- “Temporary dbspace and sbspace configuration parameters” on page 1-6
- “Dbspace and sbspace configuration parameters” on page 1-6
- “Server instance configuration parameters” on page 1-6
- “Network configuration parameters” on page 1-7
- “CPU and virtual processors configuration parameters” on page 1-7
- “AIO and buffer cleaner configuration parameters” on page 1-8
- “Locking configuration parameters” on page 1-8
- “Shared memory configuration parameters” on page 1-8
- “Checkpoint and system block configuration parameters” on page 1-9
- “Conversion guard configuration parameters” on page 1-9
- “Distributed transactions configuration parameters” on page 1-10
- “Tape device configuration parameters” on page 1-10
- “Logical log tape device configuration parameters” on page 1-10
- “Backup and restore configuration parameters” on page 1-10
- “Informix Storage Manager configuration parameters” on page 1-11
- “Data dictionary cache configuration parameters” on page 1-12
- “Data distribution configuration parameters” on page 1-12
- “User defined routine (UDR) configuration parameters” on page 1-12
- “SQL statement cache configuration parameters” on page 1-12
- “Operating system and session configuration parameters” on page 1-13
- “Index configuration parameters” on page 1-13
- “Parallel database queries configuration parameters” on page 1-13
- “Optimizer configuration parameters” on page 1-14
- “Scan configuration parameters” on page 1-15
- “SQL tracing configuration parameters” on page 1-15
- “Security configuration parameters” on page 1-15

- “Label-based access control configuration parameters” on page 1-16
- “Optical Storage subsystem configuration parameters” on page 1-16
- “Built-in character data types configuration parameters” on page 1-17
- “Sequence cache configuration parameters” on page 1-17
- “High-availability and Enterprise Replication security configuration parameters” on page 1-17
- “Enterprise Replication configuration parameters” on page 1-17
- “High-availability cluster configuration parameters” on page 1-18
- “Logical recovery threads configuration parameters” on page 1-20
- “Diagnostic configuration parameters” on page 1-20
- “Alarm program configuration parameters” on page 1-20
- “Technical support configuration parameters” on page 1-21
- “Character processing configuration parameter” on page 1-21
- “Queue and wait statistics configuration parameters” on page 1-21
- “Low memory configuration parameters” on page 1-21
- “Java configuration parameters” on page 1-22
- “Buffer pool and LRU tuning configuration parameters” on page 1-22
- “Storage provisioning configuration parameters” on page 1-22
- “MQ messaging configuration parameters” on page 1-23

Root dbspace configuration parameters

Use the following configuration parameters to configure the root dbspace.

Table 1-1. Root dbspace configuration parameters

Configuration Parameter	Reference
ROOTNAME	The root dbspace name: “ROOTNAME Configuration Parameter” on page 1-117
ROOTPATH	The path for the root dbspace: “ROOTPATH Configuration Parameter” on page 1-118
ROOTOFFSET	The offset for the root dbspace: “ROOTOFFSET Configuration Parameter” on page 1-117
ROOTSIZE	The size of the root dbspace: “ROOTSIZE configuration parameter” on page 1-118
MIRROR	Enables or disables mirroring: “MIRROR Configuration Parameter” on page 1-95
MIRRORPATH	The path for the mirrored root dbspace: “MIRRORPATH Configuration Parameter” on page 1-95
MIRROROFFSET	The offset for the mirrored root dbspace: “MIRROROFFSET Configuration Parameter” on page 1-95

Physical log configuration parameters

Use the following configuration parameters to configure physical logs.

Table 1-2. Physical log configuration parameters

Configuration Parameter	Reference
PHYSFILE	The size of the physical log: “PHYSFILE Configuration Parameter” on page 1-110

Table 1-2. Physical log configuration parameters (continued)

Configuration Parameter	Reference
PLOG_OVERFLOW_PATH	The overflow directory for physical log files: “PLOG_OVERFLOW_PATH Configuration Parameter” on page 1-110
PHYSBUFF	The size of the physical log buffer: “PHYSBUFF Configuration Parameter” on page 1-109

Logical log configuration parameters

Use the following configuration parameters to configure logical logs.

Table 1-3. Logical log configuration parameters

Configuration Parameter	Reference
LOGFILES	The number of logical log files: “LOGFILES configuration parameter” on page 1-86
LOGSIZE	The size of each logical log file: “LOGSIZE configuration parameter” on page 1-88
DYNAMIC_LOGS	The type of dynamic log allocation: “DYNAMIC_LOGS Configuration Parameter” on page 1-69
LOGBUFF	The size of the logical log buffer: “LOGBUFF configuration parameter” on page 1-85

Long transaction rollback configuration parameters

Use the following configuration parameters to control when long transactions are rolled back.

Table 1-4. Long transaction rollback configuration parameters

Configuration Parameter	Reference
LTXHWM	The percentage of the logical log files that can be filled before a long transaction is rolled back: “LTXHWM Configuration Parameter” on page 1-91
LTXEHW	The percentage of the logical log files that can be filled before the server suspends other activities so that a long transaction has exclusive use of the logs: “LTXEHW Configuration Parameter” on page 1-91

Server message file configuration parameters

Use the following configuration parameters to configure the server message file.

Table 1-5. Server message file configuration parameters

Configuration Parameter	Reference
MSGPATH	The path of the message file: “MSGPATH configuration parameter” on page 1-96
CONSOLE	The path of the console message file: “CONSOLE Configuration Parameter” on page 1-42

Tblspace tblspace in the root dbspace configuration parameters

Use the following configuration parameters to the tblspace **tblspace** in the root dbspace.

Table 1-6. Tblspace tblspace in the root dbspace configuration parameters

Configuration Parameter	Reference
TBLTBLFIRST	The first extent size for the tblspace tblspace : “TBLTBLFIRST configuration parameter” on page 1-150
TBLTBLNEXT	The next extent size for the tblspace tblspace : “TBLTBLNEXT configuration parameter” on page 1-150
TBLSPACE_STATS	Enables or disables tblspace statistics: “TBLSPACE_STATS configuration parameter” on page 1-149

Temporary dbspace and sbspace configuration parameters

Use the following configuration parameters to configure the default temporary dbspaces and sbspaces.

Table 1-7. Temporary dbspace and sbspace configuration parameters

Configuration Parameter	Reference
DBSPACETEMP	The list of dbspaces for temporary objects: “DBSPACETEMP configuration parameter” on page 1-48
SBSPACETEMP	The list of sbspaces for temporary smart large objects: “SBSPACETEMP Configuration Parameter” on page 1-123

Dbspace and sbspace configuration parameters

Use the following configuration parameters to configure the default dbspaces and sbspaces.

Table 1-8. Default dbspaces and sbspaces configuration parameters

Configuration Parameter	Reference
SBSPACENAME	The default sbspace to store smart large objects: “SBSPACENAME Configuration Parameter” on page 1-121
SYSSBSPACENAME	The default sbspace for system statistics: “SYSSBSPACENAME Configuration Parameter” on page 1-148
ONDBSPACEDOWN	Specifies the behavior of the server when a dbspace is down: “ONDBSPACEDOWN configuration parameter” on page 1-104

Server instance configuration parameters

Use the following configuration parameters to set server instance information.

Table 1-9. Server instance configuration parameters

Configuration Parameter	Reference
SERVERNUM	The unique ID for the database server instance: “SERVERNUM configuration parameter” on page 1-128

Table 1-9. Server instance configuration parameters (continued)

Configuration Parameter	Reference
DBSERVERNAME	The name of the default database server: “DBSERVERNAME configuration parameter” on page 1-47
DBSERVERALIASES	List of alternative database server names: “DBSERVERALIASES configuration parameter” on page 1-45
FULL_DISK_INIT	Prevents an accidental disk re-initialization of an existing server instance: “FULL_DISK_INIT Configuration Parameter” on page 1-80

Network configuration parameters

Use the following configuration parameters to configure the network.

Table 1-10. Network configuration parameters

Configuration Parameter	Reference
NETTYPE	The configuration of poll threads for a specific protocol: “NETTYPE Configuration Parameter” on page 1-98
LISTEN_TIMEOUT	The time the database server waits for a connection: “LISTEN_TIMEOUT Configuration Parameter” on page 1-84
MAX_INCOMPLETE_CONNECTIONS	The maximum number of incomplete connections: “MAX_INCOMPLETE_CONNECTIONS Configuration Parameter” on page 1-93
FASTPOLL	Enables or disables fast polling: “FASTPOLL Configuration Parameter” on page 1-79
NUMFDSERVERS	For network connections on UNIX, use the NUMFDSERVERS configuration parameter to specify the maximum number of poll threads to handle network connections migrating between VPs: “NUMFDSERVERS configuration parameter” on page 1-102
NS_CACHE	Defines the maximum retention time for an individual entry in the host name/IP address cache, the service cache, the user cache, and the group cache: “NS_CACHE configuration parameter” on page 1-101

CPU and virtual processors configuration parameters

Use the following configuration parameters to configure CPU virtual processors.

Table 1-11. CPU virtual processors configuration parameters

Configuration Parameter	Reference
MULTIPROCESSOR	Specifies whether the computer has multiple CPUs: “MULTIPROCESSOR Configuration Parameter” on page 1-97
VPCLASS	Configures the CPU virtual processors: “VPCLASS configuration parameter” on page 1-156

Table 1-11. CPU virtual processors configuration parameters (continued)

Configuration Parameter	Reference
VP_MEMORY_CACHE_KB	The amount of private memory blocks for the CPU virtual processors: "VP_MEMORY_CACHE_KB Configuration Parameter" on page 1-156
SINGLE_CPU_VP	Optimizes performance when there is a single CPU virtual processor: "SINGLE_CPU_VP configuration parameter" on page 1-134

AIO and buffer cleaner configuration parameters

Use the following configuration parameters to configure AIO virtual processors and buffer cleaners.

Table 1-12. AIO and buffer cleaner configuration parameters

Configuration Parameter	Reference
VPCLASS	Configures the AIO virtual processors: "VPCLASS configuration parameter" on page 1-156
CLEANERS	The number of page cleaner threads: "CLEANERS Configuration Parameter" on page 1-41
AUTO_AIOVPS	Enables or disables automatic management of AIO virtual processors: "AUTO_AIOVPS Configuration Parameter" on page 1-29
DIRECT_IO	Specifies whether to use direct I/O: "DIRECT_IO Configuration Parameter (UNIX)" on page 1-54

Locking configuration parameters

Use the following configuration parameters to set locking behavior.

Table 1-13. Locking configuration parameters

Configuration Parameter	Reference
LOCKS	The initial number of locks at startup: "LOCKS Configuration Parameter" on page 1-84
DEF_TABLE_LOCKMODE	The default table lock mode: "DEF_TABLE_LOCKMODE configuration parameter" on page 1-52

Shared memory configuration parameters

Use the following configuration parameters to configure shared memory.

Table 1-14. Shared memory configuration parameters

Configuration Parameter	Reference
RESIDENT	Controls whether shared memory is resident: "RESIDENT configuration parameter" on page 1-115
SHMBASE	The shared memory base address. Do not change this value: "SHMBASE Configuration Parameter" on page 1-129
SHMVIRTSIZE	The initial size, in KB, of the virtual segment of shared memory: "SHMVIRTSIZE configuration parameter" on page 1-132

Table 1-14. Shared memory configuration parameters (continued)

Configuration Parameter	Reference
SHMADD	The size of virtual shared memory segments: “SHMADD configuration parameter” on page 1-128
EXTSHMADD	The size of each extension shared memory segment: “EXTSHMADD Configuration Parameter” on page 1-77
SHMTOTAL	The maximum amount of shared memory for the database server: “SHMTOTAL configuration parameter” on page 1-130
SHMVIRT_ALLOCSEG	Controls when to add a memory segment: “SHMVIRT_ALLOCSEG configuration parameter” on page 1-131
SHMNOACCESS	Lists shared memory addresses that the server cannot access: “SHMNOACCESS Configuration Parameter” on page 1-130

Checkpoint and system block configuration parameters

Use the following configuration parameters to configure checkpoints, recovery time objective, and system block time.

Table 1-15. Checkpoints, recovery time objective, and system block time configuration parameters

Configuration Parameter	Reference
CKPTINTVL	How often to check if a checkpoint is needed: “CKPTINTVL configuration parameter” on page 1-41
AUTO_CKPTS	Enables or disables automatic checkpoints: “AUTO_CKPTS Configuration Parameter” on page 1-30
RTO_SERVER_RESTART	The recovery time objective for a restart after a failure: “RTO_SERVER_RESTART configuration parameter” on page 1-120
BLOCKTIMEOUT	The amount of time for a system block: “BLOCKTIMEOUT configuration parameter” on page 1-35

Conversion guard configuration parameters

Use the following configuration parameters to control information Informix uses during an upgrade to a new version of the server.

Table 1-16. Conversion guard configuration parameters

Configuration Parameter	Reference
CONVERSION_GUARD	Specifies whether to stop or continue an upgrade if an error occurs during the upgrade: “CONVERSION_GUARD configuration parameter” on page 1-42
RESTORE_POINT_DIR	Specifies where restore point files will be placed during a failed upgrade when the CONVERSION_GUARD configuration parameter is enabled: “RESTORE_POINT_DIR configuration parameter” on page 1-117

Distributed transactions configuration parameters

Use the following configuration parameters to control distributed transactions.

Table 1-17. Distributed transaction configuration parameters

Configuration Parameter	Reference
TXTIMEOUT	The distributed transaction timeout period: “TXTIMEOUT configuration parameter” on page 1-151
DEADLOCK_TIMEOUT	The maximum amount of time to wait for a lock in a distributed transaction: “DEADLOCK_TIMEOUT configuration parameter” on page 1-51
HETERO_COMMIT	Enables or disables heterogeneous commits for transactions using an EGM gateway: “HETERO_COMMIT Configuration Parameter” on page 1-81

Tape device configuration parameters

Use the following configuration parameters to configure the tape device for backups with the **ontape** utility.

Table 1-18. Tape device configuration parameters

Configuration Parameter	Reference
TAPEDEV	The tape device for backups: TAPEDEV configuration parameter
TAPEBLK	The tape block size: TAPEBLK configuration parameter
TAPESIZE	The maximum amount of data to put on one backup tape: TAPESIZE configuration parameter

Logical log tape device configuration parameters

Use the following configuration parameters to configure the tape device for logical logs with the **ontape** utility.

Table 1-19. Logical log tape device configuration parameters

Configuration Parameter	Reference
LTAPEDEV	The tape device for logical log backups: LTAPEDEV configuration parameter
LTAPEBLK	The tape block size for logical log backups: LTAPEBLK configuration parameter
LTAPESIZE	The maximum amount of data to put on one logical log backup tape: LTAPESIZE configuration parameter

Backup and restore configuration parameters

Use the following configuration parameters to control backup and restore with the ON-Bar utility. Unless specified otherwise, these configuration parameters are documented in the *IBM Informix Backup and Restore Guide*.

Table 1-20. ON-Bar configuration parameters

Configuration Parameter	Reference
BAR_ACT_LOG	The location of the ON-Bar activity log file: BAR_ACT_LOG configuration parameter
BAR_DEBUG_LOG	The location of the ON-Bar debug log file: BAR_DEBUG_LOG configuration parameter
BAR_DEBUG	The debug level for ON-Bar: BAR_DEBUG configuration parameter
BAR_MAX_BACKUP	The number of backup threads used in a backup: BAR_MAX_BACKUP configuration parameter
BAR_RETRY	The number of times to retry a backup or restore: BAR_RETRY configuration parameter
BAR_NB_XPORT_COUNT	The number of data buffers each backup process uses: BAR_NB_XPORT_COUNT configuration parameter
BAR_XFER_BUF_SIZE	The size of each data buffer: BAR_XFER_BUF_SIZE configuration parameter
RESTARTABLE_RESTORE	Enables ON-Bar to continue a backup after a failure: RESTARTABLE_RESTORE configuration parameter
BAR_PROGRESS_FREQ	How often progress messages are put in the activity log: BAR_PROGRESS_FREQ configuration parameter
BAR_BSALIB_PATH	The path for the shared library for ON-Bar and the storage manager: BAR_BSALIB_PATH configuration parameter
BACKUP_FILTER	The path of a filter program to use during backups: BACKUP_FILTER configuration parameter
RESTORE_FILTER	The path of a filter program to use during restores: RESTORE_FILTER configuration parameter
BAR_PERFORMANCE	The type of ON-Bar performance statistics to report: BAR_PERFORMANCE configuration parameter
BAR_CKPTSEC_TIMEOUT	Time in seconds to wait for an archive checkpoint to complete in the secondary server: BAR_CKPTSEC_TIMEOUT configuration parameter

Informix Storage Manager configuration parameters

Use the following configuration parameters to configure the Informix Storage Manager (ISM). These configuration parameters are documented in the *IBM Informix Storage Manager Administrator's Guide*.

Table 1-21. Informix Storage Manager configuration parameters

Configuration Parameter	Reference
ISM_DATA_POOL	The name of the ISM data pool: ISM_DATA_POOL configuration parameter
ISM_LOG_POOL	The name of the ISM log pool: ISM_LOG_POOL configuration parameter

Data dictionary cache configuration parameters

Use the following configuration parameters to configure the data dictionary caches.

Table 1-22. Data dictionary cache configuration parameters

Configuration Parameter	Reference
DD_HASHSIZE	The number of data dictionary pools: “DD_HASHSIZE Configuration Parameter” on page 1-51
DD_HASHMAX	The number of entries per pool: “DD_HASHMAX configuration parameter” on page 1-50

Data distribution configuration parameters

Use the following configuration parameters to configure the data distribution pools.

Table 1-23. Data distribution configuration parameters

Configuration Parameter	Reference
DS_HASHSIZE	The number of data distribution pools: “DS_HASHSIZE Configuration Parameter” on page 1-61
DS_POOLSIZE	The maximum number of entries in the data distribution cache: “DS_POOLSIZE Configuration Parameter” on page 1-64

User defined routine (UDR) configuration parameters

Use the following configuration parameters to configure UDRs.

Table 1-24. UDR configuration parameters

Configuration Parameter	Reference
PC_HASHSIZE	The number of hash buckets in the UDR cache: “PC_HASHSIZE Configuration Parameter” on page 1-108
PC_POOLSIZE	The maximum number of entries in the UDR cache: “PC_POOLSIZE Configuration Parameter” on page 1-109
PRELOAD_DLL_FILE	The C UDR shared library path name to load when the server starts: “PRELOAD_DLL_FILE Configuration Parameter” on page 1-112

SQL statement cache configuration parameters

Use the following configuration parameters to configure the SQL statement cache.

Table 1-25. SQL statement cache configuration parameters

Configuration Parameter	Reference
STMT_CACHE	Controls SQL statement caching: “STMT_CACHE Configuration Parameter” on page 1-142
STMT_CACHE_HITS	The number of times an SQL statement is executed before it is cached: “STMT_CACHE_HITS Configuration Parameter” on page 1-143
STMT_CACHE_SIZE	The size of the SQL statement cache: “STMT_CACHE_SIZE Configuration Parameter” on page 1-145

Table 1-25. SQL statement cache configuration parameters (continued)

Configuration Parameter	Reference
STMT_CACHE_NOLIMIT	Controls additional memory consumption of the SQL statement cache: “STMT_CACHE_NOLIMIT Configuration Parameter” on page 1-144
STMT_CACHE_NUMPOOL	The number of pools for the SQL statement cache: “STMT_CACHE_NUMPOOL Configuration Parameter” on page 1-145

Operating system and session configuration parameters

Use the following configuration parameters to configure operating system and session features.

Table 1-26. Operating system and session configuration parameters

Configuration Parameter	Reference
USEOSTIME	The precision of SQL statement timing: “USEOSTIME Configuration Parameter” on page 1-155
STACKSIZE	The size of a session stack: “STACKSIZE configuration parameter” on page 1-141
ALLOW_NEWLINE	Whether embedded new line characters are allowed in SQL statements: “ALLOW_NEWLINE Configuration Parameter” on page 1-28
USELASTCOMMITTED	Controls committed read isolation level: “USELASTCOMMITTED Configuration Parameter” on page 1-153

Index configuration parameters

Use the following configuration parameters to configure index features.

Table 1-27. Index configuration parameters

Configuration Parameter	Reference
FILLFACTOR	The percentage of index page fullness: “FILLFACTOR configuration parameter” on page 1-79
MAX_FILL_DATA_PAGES	Enables or disables filling data pages as full as possible if they have variable length rows: “MAX_FILL_DATA_PAGES Configuration Parameter” on page 1-92
BTSCANNER	Configures B-tree scanner threads: “BTSCANNER Configuration Parameter” on page 1-35
ONLIDX_MAXMEM	The amount of memory for the pre-image and updator log pools: “ONLIDX_MAXMEM Configuration Parameter” on page 1-105

Parallel database queries configuration parameters

Use the following configuration parameters to configure parallel database queries (PDQ).

Table 1-28. PDQ configuration parameters

Configuration Parameter	Reference
MAX_PDQPRIORITY	The maximum percentage of resources for a single query: "MAX_PDQPRIORITY Configuration Parameter" on page 1-94
DS_MAX_QUERIES	The maximum number of concurrent decision support queries: "DS_MAX_QUERIES Configuration Parameter" on page 1-61
DS_TOTAL_MEMORY	The maximum amount of decision support memory: "DS_TOTAL_MEMORY configuration parameter" on page 1-64
DS_MAX_SCANS	The maximum number of decision support scans: "DS_MAX_SCANS Configuration Parameter" on page 1-62
DS_NONPDQ_QUERY_MEM	The amount of non-PDQ query memory: "DS_NONPDQ_QUERY_MEM configuration parameter" on page 1-63
DATASKIP	Whether to skip a dbspace when processing a query: "DATASKIP Configuration Parameter" on page 1-43

Optimizer configuration parameters

Use the following configuration parameters to influence query execution optimizer plans and directives.

Table 1-29. Optimizer configuration parameters

Configuration Parameter	Reference
OPTCOMPIND	Controls how the optimizer determines the best query path: "OPTCOMPIND configuration parameter" on page 1-106
DIRECTIVES	Enables or disables inline optimizer directives: "DIRECTIVES configuration parameter" on page 1-55
EXT_DIRECTIVES	Enables or disables external directives: "EXT_DIRECTIVES Configuration Parameter" on page 1-76
OPT_GOAL	Controls how to optimize for fastest retrieval: "OPT_GOAL Configuration Parameter" on page 1-107
IFX_FOLDVIEW	Enables or disables folding views: "IFX_FOLDVIEW configuration parameter" on page 1-82
AUTO_REPREPARE	Enables or disables automatically re-optimizing stored procedures and re-preparing prepared statements: "AUTO_REPREPARE configuration parameter" on page 1-32
AUTO_STAT_MODE	Enables or disables the mode for selectively updating statistics for your system: "AUTO_STAT_MODE configuration parameter" on page 1-33.
STATCHANGE	Specifies a positive integer for a global percentage of a change threshold to identify data distribution statistics that need to be updated: "STATCHANGE configuration parameter" on page 1-142.
USTLOW_SAMPLE	Enables or disables the generation of index statistics based on sampling when you run UPDATE STATISTICS statements in LOW mode: "USTLOW_SAMPLE configuration parameter" on page 1-155.

Scan configuration parameters

Use the following configuration parameters to set read-ahead behavior.

Table 1-30. Scan configuration parameters

Configuration Parameter	Reference
RA_PAGES	The number of pages to read ahead: “RA_PAGES configuration parameter” on page 1-113
BATCHEDREAD_TABLE	Enables or disables light scans on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data: “BATCHEDREAD_TABLE configuration parameter” on page 1-34
BATCHEDREAD_INDEX	Enables the optimizer to automatically fetch a set of keys from an index buffer: “BATCHEDREAD_INDEX configuration parameter” on page 1-34
AUTO_READAHEAD	Changes the automatic read-ahead mode or disables or enables automatic read ahead for a query: “AUTO_READAHEAD configuration parameter” on page 1-31

SQL tracing configuration parameters

Use the following configuration parameters to set SQL tracing.

Table 1-31. SQL tracing configuration parameters

Configuration Parameter	Reference
EXPLAIN_STAT	Enables or disables including query statistics in the EXPLAIN output file: “EXPLAIN_STAT Configuration Parameter” on page 1-76
SQLTRACE	Configures SQL tracing: “SQLTRACE Configuration Parameter” on page 1-139

Security configuration parameters

Use the following configuration parameters to configure security options.

Table 1-32. Security configuration parameters

Configuration Parameter	Reference
DBCREATE_PERMISSION	Specifies users who can create databases: “DBCREATE_PERMISSION Configuration Parameter” on page 1-44
DB_LIBRARY_PATH	Specifies the locations of UDR or UDT shared libraries: “DB_LIBRARY_PATH Configuration Parameter” on page 1-44
GSKIT_VERSION	Specifies which version of IBM Global Security Kit the database server uses: “GSKIT_VERSION configuration parameter” on page 1-80
IFX_EXTEND_ROLE	Controls how to specify which users can register external routines: “IFX_EXTEND_ROLE Configuration Parameter” on page 1-82

Table 1-32. Security configuration parameters (continued)

Configuration Parameter	Reference
SECURITY_LOCALCONNECTION	Whether the database server checks the security of local connections: “SECURITY_LOCALCONNECTION Configuration Parameter” on page 1-127
UNSECURE_ONSTAT	Whether non-DBSA users can run onstat commands: “UNSECURE_ONSTAT Configuration Parameter” on page 1-152
ADMIN_USER_MODE_WITH_DBSA	Controls who can connect to the server in administration mode: “ADMIN_USER_MODE_WITH_DBSA Configuration Parameter” on page 1-26
ADMIN_MODE_USERS	Lists the users who can connect in administration mode: “ADMIN_MODE_USERS Configuration Parameter” on page 1-25
SSL_KEYSTORE_LABEL	The SSL label: “SSL_KEYSTORE_LABEL Configuration Parameter” on page 1-140
USERMAPPING	Whether mapped users can connect to Informix, and if so, whether or not the mapped user can have administrative privileges: “USERMAPPING configuration parameter” on page 1-163

Label-based access control configuration parameters

Use the following configuration parameters to configure the label-based access control (LBAC) cache. These configuration parameters are documented in the *IBM Informix Security Guide*.

Table 1-33. LBAC configuration parameters

Configuration Parameter	Reference
PLCY_HASHSIZE	The maximum number of entries in each hash bucket of the LBAC security information cache.
PLCY_POOLSIZE	The number of hash buckets in the LBAC security information cache.
USRC_HASHSIZE	The maximum number of entries in each hash bucket of the LBAC credential memory cache.
USRC_POOLSIZE	The number of hash buckets in the LBAC credential memory cache.

Optical Storage subsystem configuration parameters

Use the following configuration parameters to configure the optical storage subsystem.

Table 1-34. Optical storage subsystem configuration parameters

Configuration Parameter	Reference
STAGEBLOB	The name of the optical blob space: “STAGEBLOB Configuration Parameter” on page 1-141
OPCACHEMAX	The maximum size of the optical cache: “OPCACHEMAX Configuration Parameter (UNIX)” on page 1-106

Built-in character data types configuration parameters

Use the following configuration parameter to configure built-in character data types.

Table 1-35. Built-in character data types configuration parameters

Configuration Parameter	Reference
SQL_LOGICAL_CHAR	Enables or disables the expansion of size specifications in declarations of built-in character data types: “SQL_LOGICAL_CHAR Configuration Parameter” on page 1-138

Sequence cache configuration parameters

Use the following configuration parameter to configure the sequence cache:

Table 1-36. Sequence cache data types configuration parameters

Configuration Parameter	Reference
SEQ_CACHE_SIZE	Specifies the maximum number of sequence objects that can have preallocated values in the sequence cache: “SEQ_CACHE_SIZE configuration parameter” on page 1-127

High-availability and Enterprise Replication security configuration parameters

Use the following configuration parameters to configure security for high-availability clusters and Enterprise Replication.

Table 1-37. High-availability and Enterprise Replication security configuration parameters

Configuration Parameter	Reference
ENCRYPT_HDR	Enables or disables encryption for HDR: “ENCRYPT_HDR Configuration Parameter” on page 1-72
ENCRYPT_SMX	The level of encryption for SDS or RSS servers: “ENCRYPT_SMX Configuration Parameter” on page 1-75
ENCRYPT_CDR	The level of encryption for Enterprise Replication: .See the <i>IBM Informix Enterprise Replication Guide</i> .
ENCRYPT_CIPHERS	Lists encryption ciphers and modes: “ENCRYPT_CIPHERS Configuration Parameter” on page 1-71
ENCRYPT_MAC	The level of the message authentication code (MAC): “ENCRYPT_MAC Configuration Parameter” on page 1-73
ENCRYPT_MACFILE	The paths of MAC key files: “ENCRYPT_MACFILE configuration parameter” on page 1-74
ENCRYPT_SWITCH	The frequency to switch ciphers and keys: “ENCRYPT_SWITCH Configuration Parameter” on page 1-75

Enterprise Replication configuration parameters

Use the following configuration parameters to configure Enterprise Replication. These configuration parameters are documented in the *IBM Informix Enterprise Replication Guide*.

Table 1-38. Enterprise Replication configuration parameters

Configuration Parameter	Reference
CDR_EVALTHREADS	The numbers of evaluator threads: CDR_EVALTHREADS configuration parameter
CDR_DSLOCKWAIT	The amount of time data sync threads wait for database locks: CDR_DSLOCKWAIT configuration parameter
CDR_QUEUEMEM	The maximum amount of memory for send and receive queues: CDR_QUEUEMEM configuration parameter
CDR_NIFCOMPRESS	The network interface compression level: CDR_NIFCOMPRESS configuration parameter
CDR_SERIAL	The incremental size and starting value of serial columns: CDR_SERIAL configuration parameter
CDR_DBSPACE	The dbspace name for the syscdr database: CDR_DBSPACE configuration parameter
CDR_QHDR_DBSPACE	The name of the transaction record dbspace: CDR_QHDR_DBSPACE configuration parameter
CDR_QDATA_SBSpace	The names of sbspaces for spooled transactions: CDR_QDATA_SBSpace configuration parameter
CDR_SUPPRESS_ATSRISWARN	The data sync warnings and errors to suppress in ATS and RIS files: CDR_SUPPRESS_ATSRISWARN configuration parameter
CDR_DELAY_PURGE_DTC	The amount of time to retain delete tables: CDR_DELAY_PURGE_DTC configuration parameter
CDR_LOG_LAG_ACTION	The action taken when the database server comes close to overwriting a logical log that Enterprise Replication has not yet processed: CDR_LOG_LAG_ACTION configuration parameter
CDR_LOG_STAGING_MAXSIZE	The maximum amount of space that Enterprise Replication uses to stage log files: CDR_LOG_STAGING_MAXSIZE configuration parameter
CDR_MAX_DYNAMIC_LOGS	The maximum number of dynamic log requests Enterprise Replication can make in a session: CDR_MAX_DYNAMIC_LOGS configuration parameter

High-availability cluster configuration parameters

Use the following configuration parameters to configure high-availability clusters.

Table 1-39. High-availability cluster configuration parameters

Configuration Parameter	Reference
DRAUTO	Controls automatic failover of primary servers: “DRAUTO Configuration Parameter” on page 1-57
DRINTERVAL	The maximum interval between buffer flushes: “DRINTERVAL configuration parameter” on page 1-59
DRTIMEOUT	The network timeout period: “DRTIMEOUT configuration parameter” on page 1-60

Table 1-39. High-availability cluster configuration parameters (continued)

Configuration Parameter	Reference
DRLOSTFOUND	The path of the HDR lost-and-found file: “DRLOSTFOUND Configuration Parameter” on page 1-60
DRIDXAUTO	Enables or disables automatic index repair: “DRIDXAUTO Configuration Parameter” on page 1-59
HA_ALIAS	The server alias for a high-availability cluster: “HA_ALIAS Configuration Parameter” on page 1-81
LOG_INDEX_BUILDS	Enables or disables index page logging: “LOG_INDEX_BUILDS Configuration Parameter” on page 1-87
SDS_ENABLE	Enables or disables and SD secondary server: “SDS_ENABLE Configuration Parameter” on page 1-123
SDS_TIMEOUT	The time the primary waits for acknowledgment from an SD secondary server: “SDS_TIMEOUT Configuration Parameter” on page 1-127
SDS_TEMPDBS	The temporary dbspace used by an SD secondary server: “SDS_TEMPDBS Configuration Parameter” on page 1-125
SDS_PAGING	The paths of SD secondary paging files: “SDS_PAGING Configuration Parameter” on page 1-125
UPDATABLE_SECONDARY	Whether the secondary server can accept update, insert, or delete operations from clients: “UPDATABLE_SECONDARY Configuration Parameter” on page 1-152
FAILOVER_CALLBACK	The program called when a secondary server transitions to a standard or primary server: “FAILOVER_CALLBACK Configuration Parameter” on page 1-77
TEMPTAB_NOLOG	The default logging mode for temporary tables: “TEMPTAB_NOLOG configuration parameter” on page 1-151
DELAY_APPLY	The delay time for applying transactions on an RS secondary server: “DELAY_APPLY Configuration Parameter” on page 1-53
STOP_APPLY	Stops applying transactions on an RS secondary server: “STOP_APPLY Configuration Parameter” on page 1-145
LOG_STAGING_DIR	The directory to stage log files: “LOG_STAGING_DIR Configuration Parameter” on page 1-87
RSS_FLOW_CONTROL	“RSS_FLOW_CONTROL Configuration Parameter” on page 1-119
FAILOVER_TX_TIMEOUT	Enables or disables transaction survival behavior during failover: “FAILOVER_TX_TIMEOUT Configuration Parameter” on page 1-78
ENABLE_SNAPSHOT_COPY	Whether the server instance can be cloned using the ifxclone utility: “ENABLE_SNAPSHOT_COPY Configuration Parameter” on page 1-70
SMX_COMPRESS	The level of compression that the database server uses when sending data from the source database server to the target database server: “SMX_COMPRESS Configuration Parameter” on page 1-135

Logical recovery threads configuration parameters

Use the following configuration parameters to set logical recovery threads.

Table 1-40. Logical recovery threads configuration parameters

Configuration Parameter	Reference
ON_RECOVERY_THREADS	The number of logical recovery threads that run in parallel during a warm restore: “ON_RECOVERY_THREADS Configuration Parameter” on page 1-103
OFF_RECOVERY_THREADS	The number of logical recovery threads used in a cold restore or for fast recovery: “OFF_RECOVERY_THREADS configuration parameter” on page 1-103

Diagnostic configuration parameters

Use the following configuration parameters to control diagnostic information.

Table 1-41. Diagnostic configuration parameters

Configuration Parameter	Reference
DUMPDIR	The location of assertion failure diagnostic files: “DUMPDIR Configuration Parameter” on page 1-67
DUMPSHMEM	Controls shared memory dumps: “DUMPSHMEM Configuration Parameter (UNIX)” on page 1-68
DUMPGCORE	Enables or disables whether the database server dumps a core to the gcore file: “DUMPGCORE Configuration Parameter (UNIX)” on page 1-68
DUMPCORE	Enables or disables whether the database server dumps a core after an assertion failure: “DUMPCORE Configuration Parameter (UNIX)” on page 1-67
DUMPCNT	The maximum number of shared memory dumps for a session: “DUMPCNT Configuration Parameter (UNIX)” on page 1-66

Alarm program configuration parameters

Use the following configuration parameters to configure the alarm program.

Table 1-42. Alarm program configuration parameters

Configuration Parameter	Reference
ALARMPROGRAM	The alarm program to display event alarms: “ALARMPROGRAM configuration parameter” on page 1-27
ALRM_ALL_EVENTS	Whether the alarm program runs for all events: “ALRM_ALL_EVENTS configuration parameter” on page 1-29
STORAGE_FULL_ALARM	How often messages and events are raised when a storage space is full or a partition runs out of pages or extents: “STORAGE_FULL_ALARM Configuration Parameter” on page 1-147
SYSALARMPROGRAM	The system alarm program triggered after an assertion failure: “SYSALARMPROGRAM configuration parameter” on page 1-147

Technical support configuration parameters

The following configuration parameters to are used by technical support and are set automatically.

Table 1-43. Technical support configuration parameters

Configuration Parameter	Reference
RAS_PLOG_SPEED	Reserved for support.
RAS_LLOG_SPEED	Reserved for support.

Character processing configuration parameter

Use the following configuration parameter to control whether Informix checks if characters are valid for the locale.

Table 1-44. Character processing configuration parameter

Configuration Parameter	Reference
EILSEQ_COMPAT_MODE	Enables or disables checking character validity: “EILSEQ_COMPAT_MODE configuration parameter” on page 1-70

Queue and wait statistics configuration parameters

Use the following configuration parameters to control the collection of queue and wait statistics.

Table 1-45. Queue and wait statistics configuration parameters

Configuration Parameter	Reference
QSTATS	Enables or disables collecting queue statistics: “QSTATS Configuration Parameter” on page 1-112
WSTATS	Enables or disables collecting wait statistics: “WSTATS configuration parameter” on page 1-163

Low memory configuration parameters

Use the following configuration parameters to manage low memory.

Table 1-46. Low memory configuration parameters

Configuration Parameter	Reference
LOW_MEMORY_RESERVE	Reserves a specific amount of memory for use when critical activities are needed and the server has limited free memory: “LOW_MEMORY_RESERVE configuration parameter” on page 1-90
LOW_MEMORY_MGR	Enables you to change the default behavior of the server when it reaches the memory limit: “LOW_MEMORY_MGR configuration parameter” on page 1-89

Java configuration parameters

Use the following configuration parameters to configure Java virtual processors. These configuration parameters are documented in the *J/Foundation Developer's Guide*.

Table 1-47. Java configuration parameters

Configuration Parameter	Reference
VPCLASS	Configures Java virtual processors.
JVPPROFILE	The Java VP property file.
JVPLOGFILE	The Java VP log file.
JDKVERSION	The supported version of JDK.
JVPARGS	Configures the Java VM.
JVPCLASSPATH	The Java classpath.

Buffer pool and LRU tuning configuration parameters

Use the following configuration parameters to configure buffer pools and tune LRU queues.

Table 1-48. Buffer pool and LRU tuning configuration parameters

Configuration Parameter	Reference
BUFFERPOOL	Configures buffer pools: "BUFFERPOOL configuration parameter" on page 1-36
AUTO_LRU_TUNING	Enables or disables automatic tuning of LRU queues: "AUTO_LRU_TUNING Configuration Parameter" on page 1-30

Storage provisioning configuration parameters

Use the following configuration parameters to control information that enables the server to automatically extend or add a chunk when more space is needed in an existing storage space (dbspace, temporary dbspace, sbspace, temporary sbspace, or blobspace).

Table 1-49. Storage provisioning configuration parameters

Configuration Parameter	Reference
SP_AUTOEXPAND	Enables or disables the automatic creation or extension of chunks in a storage space: "SP_AUTOEXPAND configuration parameter" on page 1-136
SP_THRESHOLD	Defines the minimum amount of free kilobytes that can exist in a storage space: "SP_THRESHOLD Configuration Parameter" on page 1-136
SP_WAITTIME	Specifies the maximum number of seconds that a thread will wait for a storage pool to expand before returning an "out of space" error: "SP_WAITTIME Configuration Parameter" on page 1-137

MQ messaging configuration parameters

Use the following configuration parameters to configure the database server for MQ messaging. These configuration parameters are documented in the *IBM Informix Database Extensions User's Guide*.

Table 1-50. MQ configuration parameters

Configuration Parameter	Reference
MQCHLLIB	Specifies the path to the directory containing the IBM WebSphere® MQ client channel definition table: MQCHLLIB configuration parameter
MQCHLTAB	Specifies the name of IBM WebSphere the client channel definition table: MQCHLTAB configuration parameter
MQSERVER	Defines a channel, specifies the location of the IBM WebSphere MQ server, and specifies the communication method to be used: MQSERVER configuration parameter

Default Escape configuration parameter

Use the following configuration parameter as needed.

Table 1-51. Default escape configuration parameter

Configuration Parameter	Reference
DEFAULTESCCHAR	Specifies a default escape character: "DEFAULTESCCHAR configuration parameter" on page 1-53

Non-root user server installation

Use the following configuration parameters with non-root server installations.

Table 1-52. Non-root user server installation

Configuration Parameter	Reference
REMOTE_SERVER_CFG	Specifies the name of a file that lists the remote hosts that are trusted by the computer on which the database server resides. The file specified must be located in \$INFORMIXDIR/etc. For servers installed by a non-root user the file specified is used instead of the /etc/hosts.equiv file. "REMOTE_SERVER_CFG configuration parameter" on page 1-113
REMOTE_USERS_CFG	Specifies the name of a file that lists names of trusted users that exist on remote hosts. The file specified must be located in \$INFORMIXDIR/etc. For servers installed by a non-root user the file specified is used instead of the ~/.rhosts file. "REMOTE_USERS_CFG configuration parameter" on page 1-114

Table 1-52. Non-root user server installation (continued)

Configuration Parameter	Reference
S6_USE_REMOTE_SERVER_CFG	Use the S6_USE_REMOTE_SERVER_CFG configuration parameter in conjunction with the REMOTE_SERVER_CFG configuration parameter to specify the file used to authenticate secure server connections in a trusted network environment. Setting the S6_USE_REMOTE_SERVER_CFG configuration parameter to 1 specifies that the REMOTE_SERVER_CFG configuration parameter is used to authenticate secure network connections. Setting S6_USE_REMOTE_SERVER_CFG to 0 specifies that the \$INFORMIXDIR/etc/host.equiv file is used to authenticate secure network connections. "S6_USE_REMOTE_SERVER_CFG configuration parameter" on page 1-121

Additional parameters

Some configuration parameters are not in the **onconfig.std** file but you can add them to your ONCONFIG file as necessary.

Table 1-53. Parameters that you can add to an ONCONFIG file

Configuration Parameter	Reference
CHECKALLDOMAINSFORUSER	Specifies how the database server searches for user names in a networked Windows environment. "CHECKALLDOMAINSFORUSER configuration parameter" on page 1-40
DISABLE_B162428_XA_FIX	Specifies whether to free global transactions after a rollback operation. "DISABLE_B162428_XA_FIX Configuration Parameter" on page 1-56
DRDA_COMMBUFFSIZE	Specifies the size of the DRDA® communications buffer. "DRDA_COMMBUFFSIZE Configuration Parameter" on page 1-57
LIMITNUMSESSIONS	Specifies the maximum number of sessions that can connect to the database server. "LIMITNUMSESSIONS configuration parameter" on page 1-83
MSG_DATE	Inserts a date stamp at the beginning of messages printed to the online log. "MSG_DATE configuration parameter" on page 1-96
NET_IO_TIMEOUT_ALARM	Sends notification if network write operations are blocked for 30 minutes or more. "NET_IO_TIMEOUT_ALARM configuration parameter" on page 1-97

Related reference:

"onconfig.std" on page A-8

Database configuration parameter reference

You can modify parameter values to improve performance and other characteristics of the instance or database.

Each configuration parameter has one or more attributes that are relevant for that parameter. When you change the value of configuration parameters, consider the information for each parameter.

Table 1-54. Attributes for the configuration parameters

Attribute	Description
onconfig.std value	The default value that appears in the onconfig.std file. The database server uses these default values for all configurations.
<i>if not present</i>	The value that the database server supplies if the parameter is missing from your ONCONFIG file. If this value is present in onconfig.std , the database server uses the onconfig.std value. If this value is not present in onconfig.std , the database server calculates the value based on other values in onconfig.std .
<i>units</i>	The units in which the parameter is expressed.
<i>separators</i>	The separators that can be used when the parameter value has several parts. Do not use white space within a parameter value.
<i>range of values</i>	The valid values for this parameter.
<i>takes effect</i>	The time at which a change to the value of the parameter affects the operation of the database server. <i>Disk is initialized</i> means to reinitialize the database server.
<i>utilities</i>	The database server utilities that you can use to change the value of the parameter. Tip: You can use the ON-Monitor (UNIX) utility to change certain parameter values. Some of the responses for the ON-Monitor utility are Y or N (yes or no) responses. When those responses are recorded in the ONCONFIG file, Y becomes 1, and N becomes 0.
<i>refer to</i>	Cross-reference to further discussion.

In some cases you can set an environment variable or use a SET ENVIRONMENT statement in SQL to modify the database or instance behavior. You can set environment variables on the database server or in the client environment. Where you set the environment variable determines if the variable applies to all sessions or only the current session. If you set an environment variable on the database server, the environment variable applies to all sessions. If you set the environment variable in the client environment, the environment variable applies to the current session and overrides the equivalent configuration parameter (if any).

Remember: The INFORMIXDIR environment variable must always be set.

Related reference:

Appendix D, “Discontinued configuration parameters,” on page D-1

ADMIN_MODE_USERS Configuration Parameter

The ADMIN_MODE_USERS configuration parameter specifies a list of users, besides the user **informix** and members of the DBSA group, that you want to access the database server in the administration mode.

The list of users in the ADMIN_MODE_USERS configuration parameter is preserved indefinitely. Users can be removed by using the **onmode -wm** or **onmode -wf** utility.

Use the **onmode -j -U** utility to allow one or more users to access the database server in administration mode when the database is running.

You must set the **ADMIN_USER_MODE_WITH_DBSA** configuration parameter to 1 to enable the users that are listed in the **ADMIN_MODE_USERS** configuration parameter to connect to the database server in the administration mode.

onconfig.std *value*

None

range of values

Comma-separated user names, such as: Karin,Sarah,Andrew, as a string of up to 127 bytes

takes effect

When the database server is shut down and restarted

utilities

oninit -U, **onmode -j -U**, **onmode -wm**, and **onmode -wf**

refer to

- Chapter 12, “The oninit utility,” on page 12-1
- “Changing the Database Server to Administration Mode with the -j Option” on page 14-17
- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- “ADMIN_USER_MODE_WITH_DBSA Configuration Parameter”

Related reference:

“Changing the Database Server to Administration Mode with the -j Option” on page 14-17

ADMIN_USER_MODE_WITH_DBSA Configuration Parameter

The **ADMIN_USER_MODE_WITH_DBSA** configuration parameter specifies which users, besides the user **informix**, can connect to the database server in the administration mode.

onconfig.std *value*

None

if not present

0

takes effect

When the database server is shut down and restarted

range of values

Value	Valid users who can connect in the administration mode
0	<ul style="list-style-type: none">• Only the user informix can connect in the administration mode

Value	Valid users who can connect in the administration mode
1	<p>If the ADMIN_USER_MODE configuration parameter is not set, the following users can connect in the administration mode:</p> <ul style="list-style-type: none"> • The user informix • Members of the DBSA group <p>If the ADMIN_USER_MODE configuration parameter is set to a list of one or more user names, then following users can connect in the administration mode:</p> <ul style="list-style-type: none"> • The user informix • The users who have the informix group included in their group list (UNIX only) • Members of the DBSA group • The administration users that are listed in the ADMIN_MODE_USERS configuration parameter

refer to “ADMIN_MODE_USERS Configuration Parameter” on page 1-25

ALARMPROGRAM configuration parameter

Use the ALARMPROGRAM configuration parameter to display event alarms.

onconfig.std value

On UNIX: **\$INFORMIXDIR/etc/alarmprogram.sh**

On Windows: **%INFORMIXDIR%\etc\alarmprogram.bat**

if not present

On UNIX: **\$INFORMIXDIR/etc/alarmprogram.sh**

On Windows: **%INFORMIXDIR%\etc\alarmprogram.bat**

range of values

Full pathname

takes effect

When the database server is shut down and restarted

refer to

- “Writing Your Own Alarm Script” on page C-1
- *IBM Informix Backup and Restore Guide*

The following sample scripts are provided.

Script Name	Platform	Description
log_full.sh	UNIX	To back up logical logs automatically when the database server issues a log-full event alarm, set ALARMPROGRAM to log_full.sh or log_full.bat . You can modify the script and set it to the full path of ALARMPROGRAM in the onconfig file.
log_full.bat	Windows	
no_log.sh	UNIX	To disable automatic logical-log backups, set ALARMPROGRAM to no_log.sh or no_log.bat .
no_log.bat	Windows	

Script Name	Platform	Description
alarmprogram.sh	UNIX	Handles event alarms and controls logical-log backups. Modify alarmprogram.sh or alarmprogram.bat and set ALARMPROGRAM to the full pathname of alarmprogram.sh or alarmprogram.bat . See “Customizing the ALARMPROGRAM Scripts” on page C-1.
alarmprogram.bat	Windows	

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

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

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

To generate event alarms, set ALARMPROGRAM to \$INFORMIXDIR/etc/alarmprogram.sh or %INFORMIXDIR%\etc\alarmprogram.bat and modify the file according.

Important: When you choose automatic logical-log backups, backup media should always be available for the backup process.

Do not use the continuous log backup command (**onbar -b -l -C**) if you have automatic log backup setup through the ALARMPROGRAM parameter.

You can dynamically change the value of the ALARMPROGRAM configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related concepts:

Appendix C, “Event Alarms,” on page C-1

Related reference:

“ALRM_ALL_EVENTS configuration parameter” on page 1-29

ALLOW_NEWLINE Configuration Parameter

Use the ALLOW_NEWLINE configuration parameter to allow or disallow newline characters in quoted strings for all sessions.

To allow all remote sessions in a distributed query to support embedded newline characters, specify ALLOW_NEWLINE in their ONCONFIG files.

onconfig.std *value*
0

range of values

0 = Disallow the newline character in quoted strings for all sessions.

1 = Allow the newline character in quoted strings for all sessions.

takes effect

When the database server is shut down and restarted

refer to

- Quoted strings in the *IBM Informix Guide to SQL: Syntax*
- Newline characters in quoted strings in the *IBM Informix ESQL/C Programmer's Manual*

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

To allow or disallow newline characters in quoted strings for the current session when ALLOW_NEWLINE is not set, you can execute the built-in **ifx_allow_newline()** routine with 't' or 'f' as its only argument.

- 't' enables support for newline characters within quoted strings.
- 'f' has the opposite effect.

Calls to **ifx_allow_newline()** affect only the user session from which that routine is invoked.

ALRM_ALL_EVENTS configuration parameter

Use the ALRM_ALL_EVENTS configuration parameter to specify whether the ALARMPROGRAM configuration parameter runs for all events that are logged in the MSGPATH configuration parameter, or only for noteworthy events.

onconfig.std value
0

takes effect

When the database server is shut down and restarted

range of values
0, 1

If ALRM_ALL_EVENTS is set to 1, the parameter triggers the ALARMPROGRAM configuration parameter and the ALRM_ALL_EVENTS configuration parameter will display all event alarms.

Related concepts:

Appendix C, "Event Alarms," on page C-1

Related reference:

"ALARMPROGRAM configuration parameter" on page 1-27

AUTO_AIOVPS Configuration Parameter

The AUTO_AIOVPS configuration parameter enables the database server to automatically increase the number of asynchronous I/O virtual processors (AIO VPs) and page cleaner threads when the database server detects that the I/O workload has outpaced the performance of the existing AIO VPs.

onconfig.std value
1

takes effect

When the database server is shut down and restarted

range of values
0= Off
1 = On

utilities

onmode -wf or **onmode -wm**

refer to “onmode -wf, -wm: Dynamically change certain configuration parameters”
on page 14-25.

You can dynamically enable or disable the automatic increase of AIO VPs and page cleaner threads by using **onmode -wm** or **onmode -wf**.

The VP **aio** configuration parameter controls the number of AIO VPs. If the VP **aio** parameter is not set in the ONCONFIG file, the initial number of AIO VPs the database server starts when AUTO_AIOVPS is enabled is equal to the number of AIO chunks. The maximum number of AIO VPs the database server can start if VP **aio** is not set is 128.

AUTO_CKPTS Configuration Parameter

The AUTO_CKPTS configuration parameter allows the server to trigger checkpoints more frequently to avoid transaction blocking. You can dynamically enable or disable automatic checkpoints by using **onmode -wm** or **onmode -wf**.

onconfig.std *value*
1

takes effect

When the database server is shut down and restarted

range of values
0= Off 1 = On

utilities

onmode -wf or **onmode -wm**

refer to “onmode -wf, -wm: Dynamically change certain configuration parameters”
on page 14-25.

AUTO_LRU_TUNING Configuration Parameter

The AUTO_LRU_TUNING configuration parameter enables automatic LRU tuning. You can dynamically enable or disable automatic LRU tuning by using **onmode -wm** or **onmode -wf**.

onconfig.std *value*
1

takes effect

When the database server is shut down and restarted

range of values
0= Off 1 = On

utilities

onmode -wf or **onmode -wm**

refer to The following:

- “onmode -wm: Change LRU tuning status” on page 14-27

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

AUTO_READAHEAD configuration parameter

Use the AUTO_READAHEAD configuration parameter to change the automatic read-ahead mode or to disable automatic read-ahead operations for a query.

Automatic read-ahead operations help improve query performance by issuing asynchronous page requests when the database server detects that the query is encountering I/O. Asynchronous page requests can improve query performance by overlapping query processing with the processing necessary to retrieve data from disk and put it in the bufferpool.

onconfig.std *value*
1

default *value*
1

values 0 = Disable automatic read-ahead requests.

1 = Enable automatic read-ahead requests in the standard mode. The database server will automatically process read-ahead requests only when a query waits on I/O.

2 = Enable automatic read-ahead requests in the aggressive mode. The database server will automatically process read-ahead requests at the start of the query and continuously through the duration of the query.

utility commands

onmode -wm or **onmode -wf**

takes effect

When the database server is stopped and restarted or when the AUTO_READAHEAD value is changed by an **onmode -wm** or **onmode -wf** command.

Usage

Generally, the default setting (AUTO_READAHEAD = 1) is appropriate for most production environments, even cached environments. And, generally, you do not need to set AUTO_READAHEAD to 2.

While there are no specific circumstances in which aggressive read-ahead operations perform significantly better than operations performed when AUTO_READAHEAD is set to 1, aggressive read-ahead might be slightly more effective:

- For some scans that read a small amount of data
- In situations in which you switch between turning read-ahead off for small scans and on for longer scans
- For scans that only look at a small number of rows, because the server performs read-ahead operations immediately rather than waiting for the scan to encounter I/O.

For scans that might turn read-ahead operations off and on because the scan hits pockets of cached data, aggressive read-ahead operations do not turn off read-ahead operations.


You should only use aggressive read-ahead operations in situations in which you have tested both settings and know that aggressive read-ahead operations are more effective. Do not use aggressive read-ahead operations if you are not sure they are more effective.

You can use the `AUTO_READAHEAD` environment option of the `SET ENVIRONMENT` statement of SQL to enable or disable the value of the `AUTO_READAHEAD` configuration parameter for a session.

The precedence of read-ahead operations is as follows:

1. If a `SET ENVIRONMENT AUTO_READAHEAD` statement was issued for a session, the database server uses the automatic read-ahead mode specified in the statement.
2. If automatic read-ahead is enabled in the `AUTO_READAHEAD` configuration parameter and a `SET ENVIRONMENT AUTO_READAHEAD` statement was not issued, the server uses the automatic read-ahead mode specified in the `AUTO_READAHEAD` configuration parameter.
3. If the `AUTO_READAHEAD` configuration parameter is set to 0, the server performs read-ahead using information specified in the `RA_PAGES` configuration parameter. (When the `AUTO_READAHEAD` configuration parameter is enabled, the server ignores information specified in the `RA_PAGES` configuration parameter.)
4. If the `AUTO_READAHEAD` configuration parameter is set to 0 and there is no `RA_PAGES` information, the server performs passive read-ahead on 128 data pages when completing a query.

Related concepts:


 Sequential scans (Performance Guide)

 `AUTO_READAHEAD` environment option (SQL Syntax)

Related reference:

“`RA_PAGES` configuration parameter” on page 1-113

“Setting Configuration Parameters in ON-Monitor” on page 15-3

 Configure the database server to read ahead (Administrator's Guide)

“`onmode -wf, -wm`: Dynamically change certain configuration parameters” on page 14-25

AUTO_REPREPARE configuration parameter

The `AUTO_REPREPARE` configuration parameter controls whether an Informix feature is in effect that automatically re-optimizes SPL routines and re-prepares prepared objects after the schema of a table referenced by the SPL routine or by the prepared object has been changed.

By enabling the `AUTO_REPREPARE` configuration parameter, you can avoid many -710 errors and reduce the number of reprepare and reoptimize operations that you must perform manually after the schema of a table is modified.

For example, certain DDL statements modify the schema of a table, such as `CREATE INDEX`, `DROP INDEX`, `DROP COLUMN`, and `RENAME COLUMN`. If the `AUTO_REPREPARE` configuration parameter is disabled when these DDL statements are run users might receive -710 errors. These errors occur the next time that:

- You run an SPL routine that indirectly references the tables that were modified by the DDL statements.
- You run a prepared object that references the tables that were modified by the DDL statements.
-

onconfig.std *value*

1

takes effect

When the database server is shut down and restarted

range of values

- 0= Disables the automatic reparation of prepared objects after the schema of a directly or an indirectly referenced table is modified. Also disables the automatic reoptimization of SPL routines after the schema of an indirectly referenced table is modified.
- 1 = Enables the automatic reparation and automatic reoptimization feature.

refer to PREPARE, UPDATE STATISTICS, and SET ENVIRONMENT in *IBM Informix Guide to SQL: Syntax*

You can dynamically change the value of the AUTO_REPREPARE configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related concepts:

 IFX_AUTO_REPREPARE Environment Option (SQL Syntax)

AUTO_STAT_MODE configuration parameter

Use the AUTO_STAT_MODE configuration parameter to enable or disable the mode for selectively updating only stale or missing data distributions in UPDATE STATISTICS operations. You can also dynamically enable or disable this selective mode for updating statistics by using the **onmode -wm** or **onmode -wf** commands.

onconfig.std *value*

1

default value

1

range of values

0 = Disables automatic UPDATE STATISTICS operations.

1 = Enables automatic UPDATE STATISTICS operations.

takes effect

When the database server is stopped and restarted

utilities

onmode -wf or **onmode -wm**

When the AUTO_STAT_MODE configuration parameter or the AUTO_STAT_MODE session environment variable has enabled the automatic mode for selectively updating only stale or missing data distributions in UPDATE STATISTICS operations, the database server uses the value of the STATCHANGE configuration parameter to identify table or fragment distribution statistics that need to be updated.

Related concepts:

 [AUTO_STAT_MODE Environment Option \(SQL Syntax\)](#)

Related reference:

“STATCHANGE configuration parameter” on page 1-142

 [Statistics options of the CREATE TABLE statement \(SQL Syntax\)](#)

BATCHEDREAD_TABLE configuration parameter

Use the BATCHEDREAD_TABLE configuration parameter to enable or disable light scans on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data.

onconfig.std value

1

units Integer

default value

1

range of values

0 or 1

takes effect

When the database server is stopped and restarted or when the parameter is reset dynamically using the IFX_BATCHEDREAD_TABLE environment option, **onmode -wm**, or **onmode -wm**

You can use the IFX_BATCHEDREAD_TABLE environment option of the SET ENVIRONMENT statement to change the value of the BATCHEDREAD_TABLE configuration parameter for a session. The IFX_BATCHEDREAD_TABLE environment option takes precedence over the BATCHEDREAD_TABLE configuration parameter setting in the ONCONFIG file.

You can use **onmode -wm** command to change the value of BATCHEDREAD_TABLE in the shared memory to use in sessions that are created after you issue the **onmode -wm** command.

You can use **onmode -wf** to permanently update the value of the BATCHEDREAD_TABLE configuration parameter. The new value takes effect immediately and persists in the ONCONFIG file after the server restarts.

Related concepts:

 [Light scans \(Performance Guide\)](#)

Related reference:

 [SET ENVIRONMENT statement \(SQL Syntax\)](#)

BATCHEDREAD_INDEX configuration parameter

Use the BATCHEDREAD_INDEX configuration parameter to enable the optimizer to automatically fetch a set of keys from an index buffer. This reduces the number of times that a buffer is read, thus improving performance.

onconfig.std value

1

units Integer

default value

1

range of values

0 or 1

takes effect

When the database server is stopped and restarted

BLOCKTIMEOUT configuration parameter

Use the BLOCKTIMEOUT configuration parameter to specify the number of seconds that a thread or database server will hang. After the timeout, the thread or database server will either continue processing or fail.

onconfig.std *value*

3600

units Seconds

takes effect

When the database server is shut down and restarted

You can dynamically change the value of the BLOCKTIMEOUT configuration parameter by using the **onmode -wm** or **onmode -wf** command.

BTSCANNER Configuration Parameter

Use the BTSCANNER configuration parameter to set the B-tree scanner.

The B-tree scanner improves transaction processing for logged databases when rows are deleted from a table with indexes. The B-tree scanner threads remove deleted index entries and rebalance the index nodes. The B-tree scanner automatically determines which index items are to be deleted.

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

onconfig.std *value*

num=1,threshold=5000,rangesize=-1,alice=6,compression=default

syntax

BTSCANNER [num=scanner_threads][,threshold=dirty_hits][,rangesize=size]
 [,alice=mode][,compression=level]

range of values

num = The number of B-tree scanner threads to start at system startup. The default is 1.

threshold = The number of dirty hits (committed deleted index items) an index must encounter before the index is placed on the hot list for cleaning. Systems updated frequently should increase this value by a factor of 10x or 100x. The default is 5000.

rangesize = The size, in kilobytes, an index or index fragment must exceed before the index is cleaned with range scanning. To allow small indexes to be scanned by the leaf scan method set rangesize to 100. The default is OFF (-1).

alice = The mode for adaptive linear index cleaning (alice) scanning. For small- to medium-sized systems with few or no indexes above 1 gigabyte,

set alice to a mode of 6 or 7. For systems with large indexes, set alice to a higher mode. The initial system-wide alice mode determines the initial size of the bitmaps that track the deleted index entries. Valid values range from 0 to 12. The default is 0FF (0).

compression = The level at which two partially used index pages are merged. The pages are merged if the data on those pages totals a set level. Valid values for the level are low, med (medium), high, or default. The system default value is med.

takes effect

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

refer to

- “onmode -C: Control the B-tree scanner” on page 14-5.
- Configuring B-Tree Scanner Information to Improve Transaction Processing, in your *IBM Informix Performance Guide*.

BUFFERPOOL configuration parameter

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

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

onconfig.std values

Operating systems with 2K default page size:

```
BUFFERPOOL default,buffers=10000,lrus=8,lr_min_dirty=50.00,  
lr_max_dirty=60.50  
BUFFERPOOL size=2k,buffers=50000,lrus=8,lr_min_dirty=50,  
lr_max_dirty=60
```

Operating systems with 4K default page size:

```
BUFFERPOOL default,buffers=10000,lrus=8,lr_min_dirty=50.00,  
lr_max_dirty=60.50  
BUFFERPOOL size=4k,buffers=10000,lrus=8,lr_min_dirty=50,  
lr_max_dirty=60
```

syntax BUFFERPOOL default,buffers=num_buffers,lrus=num_lrus,
lr_min_dirty=percent_min,lr_max_dirty=percent_max_dirty

BUFFERPOOL
size=sizek,buffers=num_buffers,lrus=num_lrus,lr_min_dirty=percent_min
,lr_max_dirty=percent_max_dirty

takes effect

When the database server is shut down and restarted

utilities

onparams -b (See “onparams -b: Add a new buffer pool” on page 16-4.)
onspaces (See “Specifying a Non-Default Page Size with the Same Size as the Buffer Pool” on page 19-10. ON-Monitor (See Figure 15-7 on page 15-6.)

refer to “onspaces -c -d: Create a dbspace” on page 19-7 The *IBM Informix Administrator's Guide*

The BUFFERPOOL configuration parameter consists of two lines in the **onconfig.std** file, as shown in this example for a platform with a default page size of 2K:

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,lru_max_dirty=60.50
BUFFERPOOL size=2k,buffers=50000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

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

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,lru_max_dirty=60.50
BUFFERPOOL size=2k,buffers=50000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
BUFFERPOOL size=6k,buffers=3000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

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

1. The BUFFERPOOL size line, for example:
BUFFERPOOL size=2k,buffers=50000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
2. The BUFFERPOOL default line, for example:
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,lru_max_dirty=60.50
3. Database server defaults.

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

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

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

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

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

The lrus Field

onconfig.std value
lrus=8

syntax lrus=num_lrus

units Number of LRU queues

range of values

32-bit platforms: 1 through 128 64-bit platforms: 1 through 512

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

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

The buffers Field

onconfig.std value
: buffers=10000

syntax buffers=num_buffers

units Number of buffers. Each buffer is the size of the operating system page.

range of values

For 32-bit platform on UNIX: with page size equal to 2048 bytes: 100 through 1,843,200 buffers (1843200 = 1800 * 1024)

with page size equal to 4096 bytes: 100 through 921,600 buffers (921,600 = ((1800 * 1024)/4096) * 2048)

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

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

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

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

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

If your databases contain smart large objects, you need to consider them when you calculate the value for **buffers**, because smart large objects are stored in the default

page size buffer pool. If your applications frequently access smart large objects that are 2 kilobytes or 4 kilobytes in size, use the buffer pool to keep them in memory longer.

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

$$\text{Additional_BUFFERS} = \frac{\text{numcur_open_lo} * (\text{lo_userdata} / \text{pagesize})}{}$$

numcur_open_lo

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

lo_userdata

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

pagesize

is the page size in bytes for the database server.

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

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

The lru_min_dirty Field

onconfig.std *value*

lru_min_dirty=50.00

syntax lru_min_dirty=percent_min

units Percent

range of values

0 through 100 (fractional values are allowed)

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

The lru_max_dirty Field

onconfig.std *value*

lru_max_dirty=60.50

syntax lru_max_dirty=percent_max

units Percent

range of values

0 through 100 (fractional values are allowed)

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

The size Field

onconfig.std *value*
2K default page size: size=2k 4K default page size: size=4k

syntax size=size

units Kilobytes

range of values
2 through 16

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

System Page Size

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

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

Utility Description

onstat -b
Displays the system page size, given as buffer size on the last line of the output

oncheck -pr
Checks the root-dbspace reserved pages and displays the system page size in the first section of its output

ON-Monitor(UNIX)
Displays the system page size under the **Parameters > Initialize** option.
Displays system page size under the **Parameters > Shared-Memory** option, which does not require the database server to be running.

Related reference:

Appendix D, "Discontinued configuration parameters," on page D-1

CHECKALLDOMAINSFORUSER configuration parameter

Use the CHECKALLDOMAINSFORUSER configuration parameter to check all of the domains for all users.

onconfig.std *value*
Not in the **onconfig.std** file

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

range of values
0 = Disabled
1 = Enabled

Related concepts:

 Windows network domain (Administrator's Guide)

CKPTINTVL configuration parameter

Use the CKPTINTVL configuration parameter to specify the frequency, expressed in seconds, at which the database server checks to determine whether a checkpoint is needed. When a checkpoint occurs, all pages in the shared-memory buffer pool are written to disk.

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

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

onconfig.std *value*
300

units Seconds

range of values
Any value greater than or equal to 0

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

RTO_SERVER_RESTART and CKPTINTVL are mutually exclusive. If the RTO_SERVER_RESTART configuration parameter is enabled, it will trigger checkpoints and CKPTINTVL values are ignored. Otherwise, CKPTINTVL values are used to trigger checkpoints.

refer to

- Checkpoints, in the shared-memory and fast-recovery chapters of the *IBM Informix Administrator's Guide*
- Your *IBM Informix Performance Guide*

You can dynamically change the value of the CKPTINTVL configuration parameter by using the **onmode -wm** or **onmode -wf** command.

CLEANERS Configuration Parameter

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

Based on the server work load, the server automatically attempts to optimize AIO VPs and page-cleaner threads and adjust the number of AIO VPs and page-cleaner threads upward when needed. Automatic AIO VP and page-cleaner thread tuning can be disabled using the environmental variable IFX_NO_AIOVP_TUNING or the **onmode -wm** utility option.

onconfig.std *value*

8

units Number of page-cleaner threads

range of values

1 through 128

takes effect

When the database server is shut down and restarted.

utilities

onstat -F (see “**onstat -F** command: Print counts” on page 20-40.)

refer to How the database server flushes data to disk, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

CONSOLE Configuration Parameter

Use the CONSOLE configuration parameter to specify the pathname and the filename for console messages.

onconfig.std *value*

On UNIX: **\$INFORMIXDIR/tmp/online.con**

On Windows: **online.con**

range of values

Pathname

takes effect

When the database server is shut down and restarted

refer to The system console in the chapter on database server administration in the *IBM Informix Administrator's Guide*

CONVERSION_GUARD configuration parameter

Use the CONVERSION_GUARD configuration parameter to specify whether Informix stops or continues an upgrade to a new version of the server if an error occurs during the upgrade process.

onconfig.std *value*

2

units Integer

range of values

0 = Disabled.

1= Enable a restore point as part of the upgrade process, and stop the upgrade if an error related to capturing restore point data occurs.

2= Enable a restore point as part of the upgrade process, and continue the upgrade even if an error related to capturing restore point data occurs.

takes effect

When the database server is restarted

By default:

- The CONVERSION_GUARD configuration parameter is on (set to 2). If an upgrade to the new version of the server fails, you can use the **onrestorept** utility to restore your data.

- The server will store the restore point data in the `$INFORMIXDIR/tmp` directory.

If the `CONVERSION_GUARD` configuration parameter is set to 1 or 2 and the upgrade to the new version of the server fails, you can use the **onrestorept** utility to restore your data.

If the `CONVERSION_GUARD` configuration parameter is set to 2 and conversion guard operations fail (for example, because the server has insufficient space to store restore point data), and the upgrade fails, you cannot use the **onrestorept** utility to restore your data.

You can change the value of the `CONVERSION_GUARD` configuration parameter or change the directory specified in the `RESTORE_POINT_DIR` configuration parameter before starting the server that initiates an upgrade to a new version of the server. You cannot change the `CONVERSION_GUARD` or `RESTORE_POINT_DIR` values during an upgrade.

For information about the **onrestorept** utility, which you can run to undo changes made during a failed upgrade, see the *IBM Informix Migration Guide*.

DATASKIP Configuration Parameter

Use the DATASKIP configuration parameter to avoid points of media failure which can result in higher availability for your data. Setting this configuration parameter instructs the database server to skip some or all of the unavailable fragments.

Whenever the database server skips over a dbspace during query processing, a warning is returned.

syntax DATASKIP *state* [*dbspace1* *dbspace2* ...]

The *state* entry is required. If *state* is ON, at least one *dbspace* entry is required.

onconfig.std *value*
None

if not present
OFF

separators
Space

range of values
ALL = Skip all unavailable fragments. OFF = Turn off DATASKIP. ON = Skip some unavailable fragments.

utilities

onspaces -f (see “onspaces -f: Specify DATASKIP parameter” on page 19-21.) **onstat -f** (see “onstat -D command: Print page-read and page-write information” on page 20-39.)

refer to

- “onspaces -f: Specify DATASKIP parameter” on page 19-21
- The *IBM Informix Performance Guide*

ESQL/C:

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

Use the following syntax in the parameter line:

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

Usage

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

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

DBCREATE_PERMISSION Configuration Parameter

Use the DBCREATE_PERMISSION configuration parameter to restrict the permission to create databases to the user that you specify.

You can include multiple copies of the DBCREATE_PERMISSION configuration parameter in the ONCONFIG file to give additional users permission to create databases.

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

```
DBCREATE_PERMISSION informix
```

syntax DBCREATE_PERMISSION *value*

onconfig.std *value*

#DBCREATE_PERMISSION informix (suggested value, but not set)

units usernames

separator

comma

takes effect

When the database server is shut down and restarted

DB_LIBRARY_PATH Configuration Parameter

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

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

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

syntax DB_LIBRARY_PATH *value*

onconfig.std *value*
Not set

if not present

The database server can load external modules from any location

range of values

List of path names (up to 512 bytes)

separators

Comma

takes effect

When the database server is shut down and restarted

DBSERVERALIASES configuration parameter

Use the DBSERVERALIASES configuration parameter to specify an alternative name, or a list of unique alternative names for the database server. Each alias defined by DBSERVERALIASES can appear in a separate entry of the **sqlhosts** file (UNIX) or of the SQLHOSTS registry (Windows).

You can use the DBSERVERALIASES configuration parameter to specify aliases for both Secure Sockets Layer (SSL) and for non-SSL connection protocols.

If Informix supports more than one communication protocol (for example, both an IPC mechanism and the TCP network protocol), you must describe each valid connection to the database server with an entry in the **sqlhosts** file or SQLHOSTS registry. For example, suppose you have a server that has the name *sanfrancisco* defined by the DBSERVERNAME configuration parameter setting, and you set a DBSERVERALIASES value of *menlo* for an SSL connection. You must specify information for both of the *sanfrancisco* and *menlo* servers in the **sqlhosts** file or in the SQLHOSTS registry.

If the database server needs to support both the SQLI and Distributed Relational Database Architecture™ (DRDA) protocols, you must assign an alias to the DRDA database server and add an entry for this alias in the **sqlhosts** file.

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

For each alias listed in the DBSERVERALIASES configuration parameter, the database server starts an additional listener thread. If you have many client applications connecting to the database server, you can distribute the connection requests between several listener threads and reduce connection time. To take

advantage of the alternate connections, instruct some of your client applications to use a `CONNECT TO dbserveralias` statement instead of the `CONNECT TO dbservername` statement.

onconfig.std *value*

None

if not present

None

syntax DBSERVERALIASES,*name1*[-*number*],*name2*[-*number*]

The *name* variable is required.

- For **onimcsoc** or **onsoctcp** protocols, you can specify the number of multiple listen threads for the database server aliases in your **sqlhosts** information. The default value of *number* is 1.
- The server ignores the *number* value for other protocols.

separators

Comma. Blank spaces are not valid between alias entries.

range of values

At least one alias, up to a maximum of 32, separated by commas, with each alias no longer than 128 bytes. Additional aliases beyond 32 are ignored. For any line, the storage required must not exceed 512 bytes, including the comma separators and the optional *number* values.

An alias must begin with a letter and can include any printable character, except the following:

- Uppercase characters
- A field delimiter (blank space or tab)
- A newline character
- A comment character
- A hyphen or minus (= ASCII 45) character
- The @ character.

The number, ranging from 1 to 50, is for the number of multiple listener threads for **onimcsoc** or **onsoctcp** protocols.

takes effect

When the database server is shut down and restarted. In addition, you might need to update the **sqlhosts** file or registry of each database server.

Informix MaxConnect users

To use Informix MaxConnect with more than one communication protocol, specify additional database server aliases in the DBSERVERALIASES configuration parameter in the ONCONFIG file. The value of the **INFORMIXSERVER** environment variable on the client must match either the value in the DBSERVERNAME configuration parameter or one of the values in the DBSERVERALIASES configuration parameter.

For example, if you use the **onimcsoc** or **onsoctcp** protocol and you define two database server aliases named *myalias1* and *myalias2* and you intend to create four multiple listener threads for each alias, you can specify this:

DBSERVERALIASES *myalias1*-4,*myalias2*-4

Related concepts:

➡ Multiple connection types (Administrator's Guide)

Related reference:

"DBSERVERNAME configuration parameter"

➡ ONCONFIG parameters related to connectivity (Administrator's Guide)

➡ Add listen threads (Administrator's Guide)

"NETTYPE Configuration Parameter" on page 1-98

"NUMFDSERVERS configuration parameter" on page 1-102

DBSERVERNAME configuration parameter

Use the DBSERVERNAME configuration parameter to specify a unique name that you want to associate with the database server. You specify this configuration parameter when you install the database server.

The value of the DBSERVERNAME configuration parameter is called the *dbservername*. Each *dbservername* is associated with a communication protocol that is specified in the **sqlhosts** file or registry. If the database server uses multiple communication protocols, additional values for *dbservername* must be defined with the DBSERVERALIAS configuration parameter.

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

Important: To avoid conflict with other instances of Informix database servers on the same computer or node, you should use the DBSERVERNAME configuration parameter to assign a *dbservername* explicitly.

onconfig.std *value*

None

if not present

- On UNIX: *hostname*
- On Windows: *ol_hostname*

The *hostname* variable is the name of the host computer.

syntax DBSERVERNAME,*name*,[-*number*]

The *name* variable is required.

- For **onimcsoc** or **onsoctcp** protocols, you can specify the number of multiple listen threads for the database server aliases in your **sqlhosts** information. The default value of *number* is 1.
- The server ignores the *number* value for other protocols.

range of values

Up to 128 lowercase characters

DBSERVERNAME must begin with a letter and can include any printable character, except the following characters:

- Uppercase characters
- A field delimiter (space or tab)
- A newline character
- A comment character

- A hyphen, minus, or @ character

The number is number of multiple listen threads for **onimcsoc** or **onsoctcp** protocols

takes effect

When the database server is shut down and restarted. The **sqlhosts** file or registry of each database server that communicates with this database server might need to be updated. In addition, the **INFORMIXSERVER** environment variable for all users might need to be changed.

Informix MaxConnect users

The value of the **INFORMIXSERVER** environment variable on the client must match either the DBSERVERNAME or one of the entries of the DBSERVERALIASES parameter.

For example, if you have the **onimcsoc** or **onsoctcp** protocol and a database server named myserver and you want to use five multiple listen threads, you can specify:

DBSERVERNAME myserver-5


Related concepts:

 Multiple connection types (Administrator's Guide)

Related reference:

"DBSERVERALIASES configuration parameter" on page 1-45

 Connection information set in the DBSERVERNAME configuration parameter (Administrator's Guide)

 Add listen threads (Administrator's Guide)

 INFORMIXSERVER environment variable (SQL Reference)

"NETTYPE Configuration Parameter" on page 1-98

"NUMFDSERVERS configuration parameter" on page 1-102

DBSPACETEMP configuration parameter

Use the DBSPACETEMP configuration parameter to specify a list of dbspaces that the database server uses to globally manage the storage of temporary tables.

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

onconfig.std *value*

None

if not present

ROOTNAME

separators

Comma or colon (no white space)

range of values

The list of dbspaces can contain standard dbspaces, temporary dbspaces, or both. Use a colon or comma to separate the dbspaces in your list. The length of the list cannot exceed 254 bytes.

takes effect

When the database server is shut down and restarted

environment variable **DBSPACETEMP**

Specifies dbspaces that the database server uses to store temporary tables for a particular session. If **DBSPACETEMP** is not set, the default location is the **root** dbspace.

utilities

onspaces -t (see “Creating a Temporary Dbspace with the -t Option” on page 19-10.) **onstat -d flags** field (see “**onstat -d** command: Print chunk information” on page 20-34.)

refer to

- What is a temporary table, in the chapter on data storage in the *IBM Informix Administrator's Guide*
- *IBM Informix Guide to SQL: Reference*
- The order of precedence that the database server uses when it creates implicit sort files, in the *IBM Informix Performance Guide*
- The order of precedence of the default locations where the database server stores logged and unlogged temporary tables in the *IBM Informix Guide to SQL: Reference*.

DBSPACETEMP can contain dbspaces with a non-default page size, but all of the dbspaces in the DBSPACETEMP list must have the same page size. For more information about dbspaces in non-default buffer pools, see “BUFFERPOOL configuration parameter” on page 1-36.

If a client application needs to specify an alternative list of dbspaces to use for its temporary-table locations, the client can use the **DBSPACETEMP** environment variable to list them. The database server uses the storage locations that the **DBSPACETEMP** environment variable specifies only when you use the **HIGH** option of **UPDATE STATISTICS**.

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

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

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

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

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

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

You can dynamically change the value of the DBSPACETEMP configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related reference:

 DBSPACETEMP environment variable (SQL Reference)

 PSORT_DBTEMP environment variable (SQL Reference)

Use Hash Join Overflow and DBSPACETEMP

Informix uses an operating-system directory or file to direct any overflow that results from certain database operations, if you do not set the **DBSPACETEMP** environment variable or DBSPACETEMP configuration parameter.

You can specify the operating-system directory or file in the following ways:

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

Location of the sort overflow files

The following table lists the environment variables and ONCONFIG configuration parameters that you can use to specify the location of the sort overflow files.

Table 1-55. Location of sort overflow files

Variable or Parameter	Location of the sort overflow files
PSORT_DBTEMP environment variable	The location specified in the environment variable
DBSPACETEMP environment variable	The location specified in the environment variable
DBSPACETEMP configuration parameter specified in the ONCONFIG file	The dbspace that is specified in the ONCONFIG file DBSPACETEMP configuration parameter

If more than one variable or parameter is specified, the priority by which the Informix determines the location of the sort overflow files is:

1. PSORT_DBTEMP environment variable
2. DBSPACETEMP environment variable
3. DBSPACETEMP ONCONFIG variable
4. DUMPDIR
5. \$INFORMIXDIR/tmp

If the environment variables or configuration parameter are not set, the sort overflow files are placed in the **\$INFORMIXDIR/tmp** directory and the temporary tables are placed in the rootdbspace.

DD_HASHMAX configuration parameter

Use the DD_HASHMAX configuration parameter to specify the maximum number of tables in each hash bucket in the data-dictionary cache.

A *hash bucket* is the unit of storage (typically a page) whose address is computed by the hash function. A hash bucket contains several records.

For example, if the DD_HASHMAX configuration parameter is set to 10 and the DD_HASHSIZE configuration parameter is set to 59, you can store information about 590 tables in the data-dictionary cache, and each hash bucket can have a maximum of 10 tables.

onconfig.std value
10

units Maximum number of tables in a hash bucket

range of values
Positive integers

takes effect
When the database server is shut down and restarted

utilities
Use a text editor to modify the configuration file.

refer to

- Configuration effects on memory, in your *IBM Informix Performance Guide*
- “DD_HASHSIZE Configuration Parameter”

DD_HASHSIZE Configuration Parameter

Use the DD_HASHSIZE configuration parameter to specify the number of hash buckets or lists that are in the data-dictionary cache.

onconfig.std value
31

units Number of hash buckets or lists

range of values
Any positive integer; a prime number is recommended

takes effect
When the database server is shut down and restarted

utilities
Use a text editor to modify the configuration file.

refer to

- Configuration effects on memory, in your *IBM Informix Performance Guide*
- “DD_HASHMAX configuration parameter” on page 1-50

DEADLOCK_TIMEOUT configuration parameter

Use the DEADLOCK_TIMEOUT configuration parameter to specify the maximum number of seconds that a database server thread can wait to acquire a lock.

Use this parameter only for distributed queries that involve a remote database server. Do not use this parameter for nondistributed queries.

onconfig.std value
60

units Seconds

range of values
Positive integers

takes effect

When the database server is shut down and restarted

utilities

onstat -p dltouts field (See “**onstat -p** command: Print profile counts” on page 20-200.)

refer to Configuration parameters used in two-phase commits, in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*

You can dynamically change the value of the DEADLOCK_TIMEOUT configuration parameter by using the **onmode -wm** or **onmode -wf** command.

DEF_TABLE_LOCKMODE configuration parameter

Use the DEF_TABLE_LOCKMODE configuration parameter to specify the lock mode at the page or row level for new tables.

onconfig.std *value*

PAGE

if not present

PAGE

range of values

PAGE = sets lock mode to page for new tables
ROW = sets lock mode to row for new tables

takes effect

When the database server is shut down and restarted

environment variable

IFX_DEF_TABLE_LOCKMODE

refer to

- Environment variables in the *IBM Informix Guide to SQL: Reference*
- Setting lock modes, in the *IBM Informix Guide to SQL: Tutorial*
- Configuring lock mode, in the *IBM Informix Performance Guide*

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

If DEF_TABLE_LOCKMODE is set to PAGE, the USELASTCOMMITTED configuration parameter and COMMITTED READ LAST COMMITTED option of the SET ISOLATION statement cannot enable access to the most recently committed data in tables on which uncommitted transactions hold exclusive locks, unless the tables were explicitly created or altered to have ROW as their locking granularity.

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

Precedence

Command

1 (highest)

CREATE TABLE or ALTER TABLE statements that use the LOCK MODE clause

2

IFX_DEF_TABLE_LOCKMODE environment variable set on the client side

- 3 IFX_DEF_TABLE_LOCKMODE environment variable set on the server side
- 4 DEF_TABLE_LOCKMODE value in ONCONFIG file
- 5 (lowest)
Default behavior (page-level locking)

You can dynamically change the value of the DEF_TABLE_LOCKMODE configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related reference:

 IFX_DEF_TABLE_LOCKMODE environment variable (SQL Reference)

DEFAULTESCCHAR configuration parameter

The DEFAULTESCCHAR configuration parameter specifies the default escape character that is used.

onconfig.std value

The backslash character (\).

if not present

The backslash character (\) is used if no value is set in the onconfig file.

takes effect

When the database server is shut down and restarted.

range of values

Value	Description
\	The backslash character is used as the escape character.
None	No default escape character.
<i>character</i>	Any one-character value can be used as the escape character.

The default value can be overridden in a session by using the SET ENVIRONMENT DEFAULTESCCHAR statement with the escape character that you want to use. For example:

```
SET ENVIRONMENT DEFAULTESCCHAR '\'
```

Related reference:

 DEFAULTESCCHAR Environment Option (SQL Syntax)

DELAY_APPLY Configuration Parameter

Use the DELAY_APPLY configuration parameter to configure RS secondary servers to wait for a specified period of time before applying logs.

Delaying the application of log files allows you to recover quickly from erroneous database modifications by restoring the data from the RS secondary server. Set the DELAY_APPLY configuration parameter using a number from 0 to 999 followed by a modifier that indicates whether the number refers to days, hours, minutes, or seconds. When setting the value of DELAY_APPLY you must also set LOG_STAGING_DIR. If DELAY_APPLY is configured and LOG_STAGING_DIR is not set to a valid and secure directory, then the server cannot be initialized.

onconfig.std *value*

None

if not present

0S (Do not delay writing logs)

range of values (first parameter)

0 - 999 = Number of days, minutes, hours, or seconds to wait.

range of values (second parameter)

D, H, M, or S where D = Days, H = Hours, M = Minutes, and S = Seconds.

Values are not case sensitive; specify D, d, H, h, M, m, or S, s.

takes effect

when the database server is shut down and restarted or when the parameter is reset dynamically using **onmode -wf**.

utilities

onmode -wf or **onmode -wm**

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- “STOP_APPLY Configuration Parameter” on page 1-145
- “LOG_STAGING_DIR Configuration Parameter” on page 1-87
- *RS Secondary Server Latency for Disaster Recovery in the IBM Informix Administrator's Guide*

You must specify a valid and secure location for the log files. See “LOG_STAGING_DIR Configuration Parameter” on page 1-87. The logs in the staging directory are purged after the last checkpoint has been processed on the RS secondary server. Also see the “STOP_APPLY Configuration Parameter” on page 1-145.

Related reference:

“**onstat -g cluster** command: Print high-availability cluster information” on page 20-65

DIRECT_IO Configuration Parameter (UNIX)

Use the DIRECT_IO configuration parameter to control the use of direct I/O for cooked files used for dbspace chunks.

This parameter enables direct I/O (bypassing file system buffering) on UNIX platforms or concurrent IO (bypassing both file system buffering and unnecessary write serialization) on AIX® operating systems.

onconfig.std *value*

0

range of values

0 = neither direct I/O or concurrent I/O is used

1 = direct I/O, which bypasses file system buffering, is used if available

2 = concurrent I/O is enabled on AIX operating systems (The concurrent I/O option includes direct I/O and concurrent I/O.)

takes effect

When the database server is shut down and restarted

refer to

- Direct I/O and concurrent I/O information in the *IBM Informix Performance Guide*
- Direct I/O information in the *IBM Informix Administrator's Guide*
- "AUTO_AIOVPS Configuration Parameter" on page 1-29
- "**onstat -d** command: Print chunk information" on page 20-34, which displays a flag that identifies whether direct I/O, concurrent I/O, or neither is used

Direct I/O can only be used for dbspace chunks whose file systems support direct I/O for the page size.

By using direct I/O, you might be able to reduce the number of AIO virtual processors.

If direct I/O is enabled, KAIO (kernel asynchronous I/O) is used if the file system supports it. However, KAIO is not used if the environment variable KAIOOFF is set. When direct IO and KAIO are both used, the number of AIO virtual processors can be reduced. If direct IO is used, but KAIO is not, the number of AIO virtual processors should not be reduced.

Informix does not use direct or concurrent I/O for cooked files used for temporary dbspace chunks.

On AIX, if Informix uses concurrent I/O for a chunk, another program (such as an online external backup program) must also use concurrent I/O. If not, the file open operation will fail.

If Informix uses direct I/O for a chunk, and another program tries to open the chunk file without using direct I/O, the open operation will normally succeed, but there can be a performance penalty. The penalty can occur because the file system might attempt to ensure that each open operation views the same file data, either by not using direct I/O at all for the duration of the conflicting open operation, or by flushing the file system cache before each direct I/O and invalidating the file system cache after each direct write.

Direct I/O is used for dbspace chunks on Windows platforms regardless of the value of the DIRECT_IO configuration parameter.

DIRECTIVES configuration parameter

Use the DIRECTIVES configuration parameter to enable or disable the use of SQL directives.

SQL directives allow you to specify behavior for the query optimizer in developing query plans for SELECT, UPDATE, and DELETE statements.

onconfig.std *value*

1

range of values

0 optimizer directives disabled

1 optimizer directives enabled

takes effect

When the database server is shut down and restarted

environment variable

IFX_DIRECTIVES

refer to

- Environment variables in the *IBM Informix Guide to SQL: Reference*
- SQL directives, in the *IBM Informix Guide to SQL: Syntax*
- Performance impact of directives, in your *IBM Informix Performance Guide*

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

You can dynamically change the value of the DIRECTIVES configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related reference:



IFX_DIRECTIVES environment variable (SQL Reference)

DISABLE_B162428_XA_FIX Configuration Parameter

Use the DISABLE_B162428_XA_FIX configuration parameter to specify when transactions are freed.

onconfig.std *value*

Not in **onconfig.std**

units Integer

range of values

0 = (Default) Frees transactions only when an xa_rollback is called
1 = Frees transactions if transaction rollback for other than an xa_rollback

takes effect

When the database server is shut down and restarted

refer to *IBM Informix Guide to SQL: Reference*

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

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

Related reference:

 IFX_XASTDCOMPLIANCE_XAEND environment variable (SQL Reference)

DRDA_COMMBUFFSIZE Configuration Parameter

Use the DRDA_COMMBUFFSIZE configuration parameter to specify the size of the DRDA communications buffer.

When a DRDA session is established, the session is allocated a communication buffer equal to the current buffer size. If the buffer size is subsequently changed, existing connections are not affected, but new DRDA connections use the new size. Informix silently resets values greater than 2 Megabyte to 2 Megabytes and resets values less than 4 Kilobytes to the 32 Kilobyte default value.

onconfig.std *value*

Not in **onconfig.std**

if not present

32K

range of values

Minimum = 4 Kilobytes

Maximum = 2 Megabytes

takes effect

When shared memory is initialized

refer to Setting the Size of the DRDA Communications Buffer in the *IBM Informix Administrator's Guide*.

Users may specify the DRDA_COMMBUFFSIZE value in either MB or KB by adding either 'M' or 'K' to the value. The letter is case-insensitive, and the default is kilobytes. For example, a one megabyte buffer can be specified in any of these ways:

- DRDA_COMMBUFFSIZE 1M
- DRDA_COMMBUFFSIZE 1m
- DRDA_COMMBUFFSIZE 1024K
- DRDA_COMMBUFFSIZE 1024k
- DRDA_COMMBUFFSIZE 1024

DRAUTO Configuration Parameter

Use the DRAUTO configuration parameter to determine how a secondary database server reacts to a primary server failure.

This parameter should have the same value on both the primary and any secondary servers.

onconfig.std *value*

0

range of values

0 signifies OFF = Disable automatic failover.

1 signifies RETAIN_TYPE = Automatically switch the HDR secondary to standard when the primary server fails. Switch back to HDR secondary when restarting the high-availability cluster.

2 signifies REVERSE_TYPE = Automatically switch the HDR secondary to primary when the primary server fails. Switch to primary (and switch original primary to secondary) when restarting the high-availability cluster.

3 signifies that the Connection Manager performs the role of failover arbitrator.

takes effect

When shared memory is initialized

utilities

ON-Monitor > Parameters > data-Replication > Auto **onstat -g dri** command: Print high-availability data replication information” on page 20-81.)

The value you set for DRAUTO determines whether to use the built-in Informix failover arbitrator or to use the Connection Manager as the failover arbitrator. Set DRAUTO to 0 to disable automatic failover. Setting DRAUTO to 1 or 2 configures Informix to use the built-in Informix failover mechanism, which causes Informix to fail over to an HDR secondary server if the primary server fails. Setting DRAUTO to 3 allows the Connection Manager to serve as the failover arbitrator.

If the DRAUTO configuration parameter is set to 0 (OFF), and the Connection Manager failover arbitrator is not enabled, the HDR secondary database server remains a secondary database server in read-only mode when a failure of the primary server occurs.

If the DRAUTO configuration parameter is set to either 1 (RETAIN_TYPE) or 2 (REVERSE_TYPE), the HDR secondary database server automatically switches to standard (for RETAIN_TYPE) or to primary (for REVERSE_TYPE) when a failure of the primary server is detected. If the DRAUTO configuration parameter is set to RETAIN_TYPE, the original secondary database server switches back to type secondary when the HDR connection is restored. If the DRAUTO configuration parameter is set to REVERSE_TYPE, the original secondary database server switches to type primary when the HDR connection is restored, and the original primary switches to type secondary.

Setting the DRAUTO configuration parameter to 3 prevents the possibility of having multiple primary servers within a high-availability cluster. If an attempt is made to bring a server on line as a primary server and DRAUTO=3, then the Connection Manager verifies that there are no other active primary servers in the cluster. If another primary server is active, then the Connection Manager will reject the request. The server will be unavailable until the Connection Manager is enabled.

You should configure only one failover arbitrator; either the built-in Informix failover arbitrator or the Connection Manager. You should not set DRAUTO to 1 or 2 if you have configured failover using the Connection Manager.

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

DRIDXAUTO Configuration Parameter

Use the DRIDXAUTO configuration parameter to specify whether the primary High-Availability Data Replication (HDR) server automatically starts index replication if the secondary HDR server detects a corrupted index.

onconfig.std *value*
0

range of values

0 = Off

1 = On

utilities

onstat (See “**onstat -g dri** command: Print high-availability data replication information” on page 20-81.)

takes effect

When the database server is shut down and restarted

To alter the value of the DRIDXAUTO configuration parameter for an active server instance, use the **onmode -d idxauto** command. You do not need to restart the server instance. However, the **onmode -d idxauto** command will not change the value of the DRIDXAUTO configuration parameter in the ONCONFIG file. For more information, see “**onmode -d** command: Replicate an index with data-replication” on page 14-10.

DRINTERVAL configuration parameter

Use the DRINTERVAL configuration parameter to specify the maximum interval, in seconds, between flushing the high-availability data-replication buffer.

onconfig.std *value*
30

units Seconds

range of values

-1, to update synchronously

0

positive integers

takes effect

When the database server is shut down and restarted

utilities

onstat (See “**onstat -g dri** command: Print high-availability data replication information” on page 20-81.)

refer to When log records are sent, in the chapter on High-Availability Data Replication in the *IBM Informix Administrator's Guide*

You can dynamically change the value of the DRINTERVAL configuration parameter by using the **onmode -wm** or **onmode -wf** command.

DRLOSTFOUND Configuration Parameter

Use the DRLOSTFOUND configuration parameter to specify the path name to the HDR lost-and-found file. This file indicates that some transactions were committed on the HDR primary database server before that were not committed on the secondary database server when the primary database server experienced a failure.

onconfig.std *value*

On UNIX: **\$INFORMIXDIR/etc/dr.lostfound**

On Windows: **\$INFORMIXDIR\tmp**

range of values

Pathname

takes effect

When the database server is shut down and restarted

utilities

onstat (See “**onstat -g dri** command: Print high-availability data replication information” on page 20-81.)

refer to Lost-and-found transactions, in the chapter on High-Availability Data Replication in the *IBM Informix Administrator's Guide*

The DRLOSTFOUND configuration parameter is not applicable if updates between the primary and secondary database servers occur synchronously, when the DRINTERVAL configuration parameter is set to -1.

The lost-and-found file, **dr.lostfound.timestamp**, is created with a time stamp that is appended to the filename so that the database server does not overwrite another lost and found file if another file already exists. You cannot use the lost-and-found file to reapply lost transactions.

DRTIMEOUT configuration parameter

Use the DRTIMEOUT configuration parameter to specify the length of time, in seconds, that a database server in a high-availability data-replication pair waits for a transfer acknowledgment from the other database server in the pair. This parameter applies only to high-availability data-replication pairs.

onconfig.std *value*

30

units Seconds

range of values

Positive integers

takes effect

When the database server is shut down and restarted

utilities

onstat (See “**onstat -g dri** command: Print high-availability data replication information” on page 20-81.)

refer to How High-Availability Data Replication failures are detected, in the chapter on High-Availability Data Replication in the *IBM Informix Administrator's Guide*

Calculate the value for the DRTIMEOUT configuration parameter

Use the following formula to calculate the value to specify for the DRTIMEOUT configuration parameter:

$$\text{DRTIMEOUT} = \text{wait_time} / 4$$

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

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

$$\text{DRTIMEOUT} = 160 \text{ seconds} / 4 = 40 \text{ seconds}$$

You can dynamically change the value of the DRTIMEOUT configuration parameter by using the **onmode -wm** or **onmode -wf** command.

DS_HASHSIZE Configuration Parameter

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

Use the DS_HASHSIZE and DS_POOLSIZE configuration parameters to improve the performance of frequently used queries in a multiuser environment.

onconfig.std *value*
31

units Number of hash buckets or lists

range of values

Any positive integer; a prime number is recommended

takes effect

When the database server is shut down and restarted

refer to

- *IBM Informix Performance Guide* for how to monitor and tune the data-distribution cache
- “DS_POOLSIZE Configuration Parameter” on page 1-64

For information on configuration parameters for UDR cache, see the “PC_HASHSIZE Configuration Parameter” on page 1-108 and the “PC_POOLSIZE Configuration Parameter” on page 1-109.

DS_MAX_QUERIES Configuration Parameter

Use the DS_MAX_QUERIES configuration parameter to specify the maximum number of parallel database queries (PDQ) that can run concurrently.

The value of the DS_MAX_QUERIES configuration parameter is dependent on the setting for the DS_TOTAL_MEMORY configuration parameter:

- If the DS_TOTAL_MEMORY configuration parameter is set, then the value of the DS_MAX_QUERIES is $\text{DS_TOTAL_MEMORY} / 128$, rounded down to the nearest integer value.

- If the DS_TOTAL_MEMORY configuration parameter is not set, then the value of the DS_MAX_QUERIES configuration parameter is 2 * num, where num is the number of CPUs specified in the VPCLASS configuration parameter.

onconfig.std *value*
None

if not present
2 * num * 128, where num is the number of CPUs specified in the VPCLASS configuration parameter.

units Number of queries

range of values
Minimum = 1 Maximum = 8,388,608 (8 megabytes)

utilities

onmode -Q (See “onmode -D, -M, -Q, -S: Change decision-support parameters” on page 14-11.)

onstat -g mgm (See “onstat -g mgm command: Print MGM resource information” on page 20-116.)

refer to

- “VPCLASS configuration parameter” on page 1-156
- Parallel database query in your *IBM Informix Performance Guide*

The Memory Grant Manager (MGM) reserves memory for a query based on the following formula:

$$\text{memory_reserved} = \text{DS_TOTAL_MEMORY} * \frac{(\text{PDQ-priority} / 100)}{(\text{MAX_PDQPRIORITY} / 100)} *$$

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

Related reference:

 **PDQPRIORITY** environment variable (SQL Reference)

DS_MAX_SCANS Configuration Parameter

onconfig.std *value*
1048576 or (1024 * 1024)

units Number of PDQ scan threads

range of values
10 through (1024 * 1024)

utilities

onmode -S (see “onmode -D, -M, -Q, -S: Change decision-support parameters” on page 14-11.) **onstat -g mgm** (See “onstat -g mgm command: Print MGM resource information” on page 20-116.)

refer to Parallel database query in your *IBM Informix Performance Guide*

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

- The value of PDQ priority (set by the environment variable **PDQPRIORITY** or the SQL statement SET PDQPRIORITY)

- The ceiling that you set with DS_MAX_SCANS
- The factor that you set with MAX_PDQPRIORITY
- The number of fragments in the table to scan (*nfrags* in the formula)

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

$$\text{reserved_threads} = \min (nfrags, (DS_MAX_SCANS * PDQPRIORITY / 100 * MAX_PDQPRIORITY / 100))$$

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

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

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

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

Related reference:

 PDQPRIORITY environment variable (SQL Reference)

DS_NONPDQ_QUERY_MEM configuration parameter

Use the DS_NONPDQ_QUERY_MEM configuration parameter to increase the amount of memory that is available for a query that is not a Parallel Database Query (PDQ). (You can only use this parameter if PDQ priority is set to zero.)

onconfig.std
128

units Kilobytes

range of values

From 128 Kilobytes to 25 percent of the value of DS_TOTAL_MEMORY

takes effect

When the database server is initialized

utilities

onstat -g mgm (See “**onstat -g mgm** command: Print MGM resource information” on page 20-116.) **onmode** ON-Monitor

If you specify a value for the DS_NONPDQ_QUERY_MEM parameter, determine and adjust the value based on the number and size of table rows.

Tip: Set the value to generally not exceed the largest available temporary dbspace size.

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

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

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

The value for DS_NONPDQ_QUERY_MEM can be changed using the **onmode -wf** option or superseded for a session with the **onmode -wm** option. For more information about **onmode**, see “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25.

DS_POOLSIZE Configuration Parameter

onconfig.std *value*
127

default value
127

units Maximum number of entries in the data-distribution cache

range of values

Any positive value from 127 to *x*, where *x* is dependent upon the shared memory configuration and available shared memory for the server instance.

takes effect

When the database server is shut down and restarted

refer to

- IBM Informix Performance Guide for how to monitor and tune the data-distribution cache
- “DS_HASHSIZE Configuration Parameter” on page 1-61

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

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

For information on configuration parameters for UDR cache, see “PC_HASHSIZE Configuration Parameter” on page 1-108 and “PC_POOLSIZE Configuration Parameter” on page 1-109.

DS_TOTAL_MEMORY configuration parameter

Use the DS_TOTAL_MEMORY configuration parameter to specify the amount of memory available for PDQ queries. The amount should be smaller than the computer physical memory, minus fixed overhead such as operating-system size and buffer-pool size.

onconfig.std *value*

None

if not present

If SHMTOTAL=0 and DS_MAX_QUERIES is set, DS_TOTAL_MEMORY = DS_MAX_QUERIES * 128.

If SHMTOTAL=0 and DS_MAX_QUERIES is not set, DS_TOTAL_MEMORY = num_cpu_vps * 2 * 128.

units Kilobytes

range of values

If DS_MAX_QUERY is set, the minimum value is DS_MAX_QUERY * 128.

If DS_MAX_QUERY is not set, the minimum value is num_cpu_vps * 2 * 128.

There is no maximum value limit other than any limit that you might have with the software that you use on your machine.

utilities

onmode -M (see “onmode -D, -M, -Q, -S: Change decision-support parameters” on page 14-11.) **onstat -g mgm** (See “onstat -g mgm command: Print MGM resource information” on page 20-116.)

refer to

- Your *IBM Informix Performance Guide* for the algorithms
- “SHMTOTAL configuration parameter” on page 1-130
- “SHMVIRTSIZE configuration parameter” on page 1-132
- “VPCLASS configuration parameter” on page 1-156
- The maximum memory available on your platform, in the machine notes

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

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

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

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

Algorithm for DS_TOTAL_MEMORY

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

AUTO_STAT_MODE configuration parameter

Use the AUTO_STAT_MODE configuration parameter to enable or disable the mode for selectively updating only stale or missing data distributions in UPDATE STATISTICS operations. You can also dynamically enable or disable this selective mode for updating statistics by using the **onmode -wm** or **onmode -wf** commands.

onconfig.std *value*

1

default value

1

range of values

0 = Disables automatic UPDATE STATISTICS operations.

1 = Enables automatic UPDATE STATISTICS operations.

takes effect

When the database server is stopped and restarted

utilities

onmode -wf or **onmode -wm**

When the AUTO_STAT_MODE configuration parameter or the AUTO_STAT_MODE session environment variable has enabled the automatic mode for selectively updating only stale or missing data distributions in UPDATE STATISTICS operations, the database server uses the value of the STATCHANGE configuration parameter to identify table or fragment distribution statistics that need to be updated.

Related concepts:

 [AUTO_STAT_MODE Environment Option \(SQL Syntax\)](#)

Related reference:

“STATCHANGE configuration parameter” on page 1-142

 [Statistics options of the CREATE TABLE statement \(SQL Syntax\)](#)

DUMPCNT Configuration Parameter (UNIX)

onconfig.std *value*

1

if not present

1

units Number of assertion failures

range of values

Positive integers

takes effect

When the database server is shut down and restarted

utilities

onmode -wf or **onmode -wm**

refer to

- Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- “DUMPDIR Configuration Parameter”
- “DUMPSHMEM Configuration Parameter (UNIX)” on page 1-68

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

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

DUMPCORE Configuration Parameter (UNIX)

onconfig.std value
0

range of values
0 = Do not dump core image. 1 = Dump core image.

takes effect
When the database server is shut down and restarted

refer to Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

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

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

DUMPDIR Configuration Parameter

onconfig.std value
On UNIX: **\$INFORMIXDIR/tmp** On Windows: **\$INFORMIXDIR\tmp**

if not present
\$INFORMIXDIR/tmp

range of values
Any directory to which user **informix** has write access

takes effect
When the database server is shut down and restarted

refer to

- Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*
- “DUMPCNT Configuration Parameter (UNIX)” on page 1-66
- “DUMPSHMEM Configuration Parameter (UNIX)” on page 1-68

DUMPDIR specifies a directory in which the database server dumps shared memory, **gcore** files, or messages from a failed assertion. Because shared memory

can be large, set DUMPPDIR to a file system with a significant amount of space. The directory to which DUMPPDIR is set must exist for the server to start.

DUMPGCORE Configuration Parameter (UNIX)

onconfig.std *value*
0

range of values

0 = Do not dump **gcore**. 1 = Dump **gcore**.

takes effect

When the database server is shut down and restarted

refer to Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

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

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

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

DUMPSHMEM Configuration Parameter (UNIX)

onconfig.std *value*
1

range of values

0 = Do not create a shared memory dump 1 = Create a shared memory dump of all the shared memory that the database uses 2 = Create a shared memory dump that excludes the buffer pool in the resident memory

takes effect

When the database server is shut down and restarted

utilities

onmode -wf or **onmode -wm**

refer to

- Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*
- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- “**onstat -o** command: Output shared memory contents to a file” on page 20-198

- “Running **onstat** Commands on a Shared Memory Dump File” on page 20-25
- “DUMPCNT Configuration Parameter (UNIX)” on page 1-66
- “DUMPPDIR Configuration Parameter” on page 1-67

DUMPSHMEM indicates whether a shared memory dump is created on an assertion failure and how much memory is written to the **shmем.рid.cnt** file in the directory specified by the DUMPPDIR configuration parameter. If DUMPSHMEM is set to 1, all the shared memory that the database server uses is dumped, which can result in a large file. When space is limited, set DUMPSHMEM to 2 because this setting creates a smaller shared-memory dump file.

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

DYNAMIC_LOGS Configuration Parameter

onconfig.std *value*
2

if not present
2 (Default)

range of values

0 = Turn off dynamic-log allocation.

1 = Set off the “log file required” alarm and pause to allow manual addition of a logical-log file. You can add a log file immediately after the current log file or to the end of the log file list.

2 = Turn on dynamic-log allocation. When the database server dynamically adds a log file, it sets off the “dynamically added log file” alarm.

takes effect

For HDR: when the database server is shut down and restarted

For Enterprise Replication: when Enterprise Replication is started

utilities

“onparams -a -d *dbspace*: Add a logical-log file” on page 16-2

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- “LTXEHWМ Configuration Parameter” on page 1-91
- “LTXHWM Configuration Parameter” on page 1-91
- Logical logs in the *IBM Informix Administrator’s Guide*

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

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

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

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

EILSEQ_COMPAT_MODE configuration parameter

Use the EILSEQ_COMPAT_MODE configuration parameter to control if Informix checks whether character data inserted by a client application contains code point sequences not recognized by the current locale.

onconfig.std *value*
0

if not present value
0

range of values

0 = Informix validates incoming character sequences with the current locale and returns error -202 if any characters are not valid.

1 = Informix does not validate incoming character sequences.

takes effect

When the database server is shut down and restarted

For example, if the EILSEQ_COMPAT_MODE configuration parameter is set to 0, an ESQL/C program that attempts to execute the following SQL statement on a database created with the **DB_LOCALE** environment variable set to **en_US.utf8** receives error -202 because the sequence 0x369 is not a valid byte sequence in UTF-8:

```
INSERT INTO x (columnA) VALUES '\x03\x69|data';
```

If you are using UTF-8 databases, set the **GL_USEGLU** environment variable to 1 on the client to enable the server to recognize a wider range of Unicode characters instead of setting the EILSEQ_COMPAT_MODE configuration parameter to 1 to prevent errors.

Related reference:

 The GL_USEGLU environment variable (GLS User's Guide)

ENABLE_SNAPSHOT_COPY Configuration Parameter

Use the ENABLE_SNAPSHOT_COPY configuration parameter to enable or disable the ability to clone a server using the **ifxclone** utility.

The ENABLE_SNAPSHOT_COPY configuration parameter determines whether you can create a clone of a server using the **ifxclone** utility. Set the ENABLE_SNAPSHOT_COPY configuration parameter to 1 to allow cloning. Set the value to 0 to prohibit cloning the server using the **ifxclone** utility.

onconfig.std *value*

None

if not present

0 (Do not allow the server to be cloned)

range of values

0, 1 (0 = prohibit clone, 1 = permit clone)

takes effect

when the database server is shut down and restarted or when the parameter is reset dynamically using **onmode -wf**.

utilities

onmode -wf or **onmode -wm**

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- Chapter 18, “The ifxclone utility,” on page 18-1

Related reference:

“onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

Chapter 18, “The ifxclone utility,” on page 18-1

ENCRYPT_CIPHERS Configuration Parameter

onconfig.std *value*

None

syntax ENCRYPT_CIPHERS all|allbut:<list of ciphers and modes>|cipher:mode{,cipher:mode ...}

- all

Specifies to include all available ciphers and modes, except ECB mode.
For example: ENCRYPT_CIPHERS all

- allbut:<list of ciphers and modes>

Specifies to include all ciphers and modes except the ones in the list.
Separate ciphers or modes with a comma. For example: ENCRYPT_CIPHERS allbut:<cbc,bf>

- cipher:mode

Specifies the ciphers and modes. Separate cipher-mode pairs with a comma. For example: ENCRYPT_CIPHERS des3:cbc,des3:ofb

takes effect

For HDR: when the database server is shut down and restarted

For Enterprise Replication: when Enterprise Replication is started

refer to

- “ENCRYPT_HDR Configuration Parameter” on page 1-72
- “ENCRYPT_MAC Configuration Parameter” on page 1-73
- “ENCRYPT_MACFILE configuration parameter” on page 1-74
- “ENCRYPT_SWITCH Configuration Parameter” on page 1-75
- HDR Encryption Options in the *IBM Informix Administrator's Guide*
- Using High-Availability Data Replication in the *IBM Informix Enterprise Replication*

The ENCRYPT_CIPHERS configuration parameter defines all ciphers and modes that can be used by the current database session. ENCRYPT_CIPHERS is used for Enterprise Replication and High-Availability Data Replication only.

The cipher list for **allbut** can include unique, abbreviated entries. For example, **bf** can represent bf-1, bf-2, and bf-3; however, if the abbreviation is the name of an actual cipher, then only that cipher is eliminated. Therefore, **des** eliminates only the des cipher, but **de** eliminates des, des3, and desx.

Important: The encryption cipher and mode used is randomly chosen among the ciphers common between the two servers. It is strongly recommended that you do not specify specific ciphers. For security reasons, all ciphers should be allowed. If a specific cipher is discovered to have a weakness, then that cipher can be eliminated by using the **allbut** option.

The following ciphers are supported. For an updated list, see the Release Notes.

Cipher	Description
des	DES (64-bit key)
des3	Triple DES
desx	Extended DES (128-bit key)
aes	AES 128bit key
aes192	AES 192bit key
bf-1	Blow Fish (64-bit key)
bf-2	Blow Fish (128-bit key)
bf-3	Blow Fish (192-bit key)
aes128	AES 128bit key
aes256	AES 256bit key

The following modes are supported.

ecb Electronic Code Book (ECB)
cbc Cipher Block Chaining
cfb Cipher Feedback
ofb Output Feedback

Because **cdb** mode is considered weak, it is only included if specifically requested. It is not included in the **all** or the **allbut** list.

All ciphers support all modes, except the **desx** cipher, which only supports the **cbc** mode.

ENCRYPT_HDR Configuration Parameter

onconfig.std *value*
None

range of values
0 = Disables HDR encryption
1 = Enables HDR encryption

takes effect

when the server is initialized

refer to

- “ENCRYPT_CIPHERS Configuration Parameter” on page 1-71
- “ENCRYPT_MAC Configuration Parameter”
- “ENCRYPT_MACFILE configuration parameter” on page 1-74
- “ENCRYPT_SWITCH Configuration Parameter” on page 1-75
- HDR Encryption Options in the *IBM Informix Administrator's Guide*
- Using High-Availability Data Replication in the IBM Informix Enterprise Replication

ENCRYPT_HDR enables or disables HDR encryption. Enabling HDR encryption provides a secure method for transferring data from one server to another in an HDR pair. HDR encryption works in conjunction with Enterprise Replication (ER) encryption. However, it is not necessary to have ER encryption enabled for HDR encryption. HDR encryption works whether ER encryption is enabled or not. HDR and ER share the same encryption configuration parameters: ENCRYPT_CIPHERS, ENCRYPT_MAC, ENCRYPT_MACFILE and ENCRYPT_SWITCH.

ENCRYPT_MAC Configuration Parameter

Use the ENCRYPT_MAC configuration parameter to control the level of message authentication code (MAC) generation. This configuration parameter is used only for Enterprise Replication and High-Availability Data Replication.

onconfig.std *value*

None

range of values

One or more of the following options, separated by commas:

Option	Description
off	Does not use MAC generation
low	Uses XOR folding on all messages
medium	Uses SHA1 MAC generation for all messages that are greater than 20 bytes long and XOR folding on smaller messages
high	Uses SHA1 MAC generation on all messages.

example

ENCRYPT_MAC medium,high

takes effect

For HDR: when the database server is shut down and restarted

For Enterprise Replication: when Enterprise Replication is started

refer to

- “ENCRYPT_CIPHERS Configuration Parameter” on page 1-71
- “ENCRYPT_HDR Configuration Parameter” on page 1-72
- “ENCRYPT_MACFILE configuration parameter” on page 1-74
- “ENCRYPT_SWITCH Configuration Parameter” on page 1-75
- HDR Encryption Options in the *IBM Informix Administrator's Guide*

- Using High-Availability Data Replication in the IBM Informix Enterprise Replication

The level is prioritized to the highest value. For example, if one node has a level of **high** and **medium** enabled and the other node has only **low** enabled, then the connection attempt fails. Use the **off** entry between servers only when a secure network connection is guaranteed.

ENCRYPT_MACFILE configuration parameter

Use the ENCRYPT_MACFILE configuration parameter to specify a list of the full path names of MAC key files. This configuration parameter is used only for Enterprise Replication and High-Availability Data Replication.

onconfig.std *value*
None

units pathnames, up to 1536 bytes in length

range of values

One or more full path and filenames separated by commas, and the optional **builtin** keyword. For example: ENCRYPT_MACFILE /usr/local/bin/mac1.dat, /usr/local/bin/mac2.dat,builtin

takes effect

For HDR: when the database server is shut down and restarted.

For Enterprise Replication: when Enterprise Replication is started.

refer to

- “ENCRYPT_CIPHERS Configuration Parameter” on page 1-71
- “ENCRYPT_HDR Configuration Parameter” on page 1-72
- “ENCRYPT_MAC Configuration Parameter” on page 1-73
- “ENCRYPT_SWITCH Configuration Parameter” on page 1-75
- HDR Encryption Options in the *IBM Informix Administrator's Guide*
- Using High-Availability Data Replication in the IBM Informix Enterprise Replication guide

Each of the entries for the ENCRYPT_MACFILE configuration parameter is prioritized and negotiated at connect time. The prioritization for the MAC key files is based on their creation time by the **GenMacKey** utility. The entry created from the **builtin** keyword has the lowest priority. Because the MAC key files are negotiated, you should periodically change the keys.

Specify the builtin keyword

Use the **builtin** keyword to specify the built-in key. The **builtin** keyword provides limited message verification (some validation of the received message and determination that it appears to have come from an Informix client or server). The strongest verification is done by a site-generated MAC key file.

Generate a MAC key file

1. Execute the following command from the command line:

GenMacKey -o filename

The *filename* is the name of the MAC key file.

2. Update the ENCRYPT_MACFILE configuration parameter on participating servers to include the location of the new MAC key file.
3. Distribute the new MAC key file.

ENCRYPT_SMX Configuration Parameter

Use the ENCRYPT_SMX configuration parameter to set the level of encryption for high-availability configurations on secondary servers.

onconfig.std *value*
None

range of values

Value	Explanation
0	Off. Do not encrypt.
1	On. Encrypt where possible. Encrypt SMX transactions when the database server being connected to also supports encryption.
2	On. Always encrypt. Only connections to encrypted database servers are allowed.

takes effect

When the server is restarted

refer to

- Server Multiplexer group (SMX) Connections in the *IBM Informix Administrator's Guide*
- Using High-Availability Data Replication in the IBM Informix Enterprise Replication guide

ENCRYPT_SWITCH Configuration Parameter

Use the ENCRYPT_SWITCH configuration parameter to define the frequency at which ciphers or secret keys are renegotiated. This configuration parameter is used only for Enterprise Replication and High-Availability Data Replication.

The longer the secret key and encryption cipher remains in use, the more likely the encryption rules might be broken by an attacker. To avoid this, cryptologists recommend changing the secret keys on long-term connections. The default time that this renegotiation occurs is once an hour.

onconfig.std *value*
None

syntax ENCRYPT_SWITCH *cipher_switch_time*, *key_switch_time*

- *cipher_switch_time* specifies the minutes between cipher renegotiation
- *key_switch_time* specifies the minutes between secret key renegotiation

units minutes

range of values
positive integers

takes effect

For HDR: when the database server is shut down and restarted

For Enterprise Replication: when Enterprise Replication is started

refer to

- “ENCRYPT_CIPHERS Configuration Parameter” on page 1-71
- “ENCRYPT_HDR Configuration Parameter” on page 1-72
- “ENCRYPT_MAC Configuration Parameter” on page 1-73
- “ENCRYPT_MACFILE configuration parameter” on page 1-74
- HDR Encryption Options in the *IBM Informix Administrator’s Guide*
- Using High-Availability Data Replication in the *IBM Informix Enterprise Replication guide*

EXPLAIN_STAT Configuration Parameter

Use the EXPLAIN_STAT configuration parameter to enable or disable the inclusion of a Query Statistics section in the explain output file.

You can generate the output file by using either the SET EXPLAIN statement or the **onmode -Y sessionid** command. When you enable the EXPLAIN_STAT configuration parameter, the Query Statistics section shows the estimated number of rows and the actual number of returned rows in the Query Plan.

onconfig.std *value*
1

range of values
0 to 1

takes effect
When the database server is shut down and restarted

refer to

- “onmode -Y: Dynamically change SET EXPLAIN” on page 14-28
- SET EXPLAIN, in the *IBM Informix Guide to SQL: Syntax*
- Query Plan Report, in the *IBM Informix Performance Guide*

Value	Description
0	Disables the display of query statistics.
1	Enables the display of query statistics.

EXT_DIRECTIVES Configuration Parameter

onconfig.std *value*
0

range of values
0, 1, 2

takes effect
When the database server is shut down and restarted

environment variable
IFX_EXTDIRECTIVES

refer to

- Environment variables and information about the **sysdirectives** system catalog table, in the *IBM Informix Guide to SQL: Reference*
- Query optimizer directives, in the *IBM Informix Guide to SQL: Syntax*
- SET ENVIRONMENT EXTDIRECTIVES statement of SQL, in the *IBM Informix Guide to SQL: Syntax*

- Using external optimizer directives, in the *IBM Informix Performance Guide*

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

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

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

The setting specified by the **SET ENVIRONMENT EXTDIRECTIVES** statement of SQL overrides (for the current user session only) the settings of both the **IFX_EXTDIRECTIVES** environment variable and of the **EXT_DIRECTIVES** configuration parameter.

Related tasks:

[EXTDIRECTIVES Environment Option \(SQL Syntax\)](#)

Related reference:

[IFX_EXTDIRECTIVES environment variable \(SQL Reference\)](#)

EXTSHMADD Configuration Parameter

onconfig.std *value*
8192

range of values
1024 through 524,288

units Kilobytes

takes effect
When the database server is shut down and restarted

utilities
onstat -g seg

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

FAILOVER_CALLBACK Configuration Parameter

onconfig.std *value*
None

default value
None

range of values

Pathname

takes effect

When the database server is shut down and restarted

The database server executes the script specified by `FAILOVER_CALLBACK` when a database server transitions from a secondary server to a primary or standard server. Set `FAILOVER_CALLBACK` to the full pathname of the script.

FAILOVER_TX_TIMEOUT Configuration Parameter

In high-availability cluster environments, use the `FAILOVER_TX_TIMEOUT` configuration parameter to enable transactions to complete after failover of the primary server.

Use the `FAILOVER_TX_TIMEOUT` configuration parameter to indicate the maximum number of seconds after failover that the server waits before rolling back transactions. Set the `FAILOVER_TX_TIMEOUT` configuration parameter to the same value on all servers in a high-availability cluster.

onconfig.std *value*

0

syntax `FAILOVER_TX_TIMEOUT seconds`

takes effect

When the database server is stopped and restarted

utilities

onmode -wf or **onmode -wm**

by default

Transaction survival is disabled by default.

When a failover occurs in a high-availability cluster environment, one of the secondary servers takes over the role of the primary server. The secondary server that becomes the new primary server is called the *failover server*.

You enable transaction survival by setting the `FAILOVER_TX_TIMEOUT` configuration parameter to a value greater than zero. When transaction survival is enabled, the failover server must be able to contact the remaining secondary servers to synchronize and resume any open transactions. Similarly, the surviving secondary servers must be able to establish connections to the failover server to re-send any pending transactions. The `FAILOVER_TX_TIMEOUT` configuration parameter specifies how long the servers will wait before rolling back transactions.

On the failover server, if the number of seconds specified by `FAILOVER_TX_TIMEOUT` has expired, any open transactions that could not be synchronized with a surviving server are terminated and rolled back.

On the remaining secondary servers, if the number of seconds specified by `FAILOVER_TX_TIMEOUT` has expired, any open transactions on that server return an error.

Set `FAILOVER_TX_TIMEOUT` to 0 to immediately roll back all open transactions when failover occurs.

Limitations

If the primary server fails and a secondary server fails to take over the role of the primary server, then any open transactions are rolled back, and the client will be unable to perform any additional update activity. For example, suppose an update activity has been started on a secondary server and the primary server fails. Now suppose that failover processing does not complete and a new primary server is not established. After a predetermined amount of time, the client request (that is, the sqlexec thread) times out, placing the sqlexec thread in an indeterminate state.

In the preceding scenario, active transactions are rolled back, but the physical rollback cannot occur until the new primary server is established (because the primary server manages the logs). Under these circumstances, the session may be unaware of operations that have been performed on the secondary server. The session may be unaware of the rollback of a partially applied transaction because the rollback of the partial transaction cannot occur until a new primary server is established.

If your high-availability cluster contains an HDR secondary server, the DRINTERVAL configuration parameter must be set to -1 to enable transactions to complete after failover of the primary server. Setting DRINTERVAL to -1 on both the primary server and the HDR secondary server configures synchronous updates between the servers.

FASTPOLL Configuration Parameter

onconfig.std *value*

1

range of values

0 = Disables fast polling. 1 = Enables fast polling.

takes effect

When the database server is shut down and restarted

Use the FASTPOLL configuration parameter to enable or disable fast polling of your network. FASTPOLL is a platform-specific configuration parameter.

FILLFACTOR configuration parameter

Use the FILLFACTOR configuration parameter to specify the degree of index-page fullness. A low value provides room for growth in the index. A high value compacts the index.

If an index is full (100 percent), any new inserts result in splitting nodes. You can also set the FILLFACTOR as an option on the CREATE INDEX statement. The setting on the CREATE INDEX statement overrides the ONCONFIG file value.

You cannot use the FILLFACTOR configuration parameter with a forest of trees index.

onconfig.std *value*

90

units Percent

range of values

1 through 100

takes effect

When the index is built. Existing indexes are not changed. To use the new value, the indexes must be rebuilt.

refer to "Structure of B-Tree Index Pages" on page 4-16

You can dynamically change the value of the FILLFACTOR configuration parameter by using the **onmode -wm** or **onmode -wf** command.

FULL_DISK_INIT Configuration Parameter

Use the FULL_DISK_INIT configuration parameter to prevent an accidental disk reinitialization of an existing database server instance. This configuration parameter specifies whether or not the disk initialization command (**oninit -i**) can run on your Informix instance when a page zero exists at the root path location, which is at the first page of the first chunk location.

onconfig.std *value*
0

default value
0

takes effect

When the database server is shut down and restarted

range of values

0= The **oninit -i** command runs only if there is not a page zero at the root path location.

1 = The **oninit -i** command runs under all circumstances, but also resets the FULL_DISK_INIT configuration parameter to 0 after the disk initialization.

When the FULL_DISK_INIT configuration parameter is set to 1, any instance startup command (for example, **oninit** as well as **oninit -i**) resets the configuration parameter to 0.

If you start to run the **oninit -i** command when the FULL_DISK_INIT configuration parameter is set to 0 and the database server finds a page zero, the **oninit -i** command does not run and the server reports an error in the **online.log**.

Page zero is the Informix system page that contains general information about the server instance. This page is created when the server instance is initialized.

Related reference:

Chapter 12, "The oninit utility," on page 12-1

GSKIT_VERSION configuration parameter

Use the GSKIT_VERSION configuration parameter to specify the major version of IBM Global Security Kit (GSKit) that the database server uses for encryption and SSL communication.

If the database server is used with other IBM products on the same computer, and a different version of GSKit has been installed with one of the other IBM products, the database server can be configured to use the different version of GSKit. By default, the database server uses GSKit 8. You can set the GSKIT_VERSION parameter to use GSKit 7 or a later version of GSKit.

onconfig.std *value*

0

units Positive integer

range of values

7 to The latest major release of GSKit

takes effect

During database initialization

refer to

- The *IBM Informix Security Guide*

Related concepts:



Secure sockets layer protocol (Security Guide)



IBM Global Security Kit (Security Guide)

HA_ALIAS Configuration Parameter

onconfig.std *value*

None

takes effect

When the database server is shut down and restarted

The value specified by the HA_ALIAS configuration parameter must be one of the values specified in the DBSERVERNAME or DBSERVERALIASES configuration parameters, and must refer to a server whose connection type is a TCP network protocol.

The value specified by the HA_ALIAS configuration parameter is optional and represents the name by which the server is known within a high-availability cluster. Valid values are the same as for DBSERVERNAME.

When a secondary server connects to a primary server, the secondary server sends the name of a network alias that can be used in case of failover. The setting of HA_ALIAS is used to describe which network alias will be sent.

HETERO_COMMIT Configuration Parameter

onconfig.std *value*

0

range of values

1 = Enable heterogeneous commit. 0 = Disable heterogeneous commit.

takes effect

When the database server is shut down and restarted

refer to

- Heterogeneous commit protocol, in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*
- *IBM Informix Enterprise Gateway Manager User Manual*

The HETERO_COMMIT configuration parameter specifies whether or not the database server is prepared to participate with IBM Informix Gateway products in heterogeneous commit transactions. Setting HETERO_COMMIT to 1 allows a

single transaction to update *one* non-Informix database (accessed with any of the Gateway products) and one or more Informix databases.

If HETERO_COMMIT is 0, a single transaction can update databases as follows:

- One or more Informix databases and no non-Informix databases
- One non-Informix database and no Informix databases

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

IFX_EXTEND_ROLE Configuration Parameter

onconfig.std *value*

1

range of values

1 or On (default) = Enables the requirement for the EXTEND role so that administrators can grant privileges to a user to create or drop a UDR that includes the EXTERNAL clause.

0 or Off = Disables the requirement for the EXTEND role, so that any user who holds the USAGE ON LANGUAGE privilege for the appropriate external language (C or JAVA) can register or drop an external routine that was written in that language.

refer to Information on security for external routines in the *IBM Informix Security Guide*

Your database system administrator (DBSA), by default user **informix**, can use the IFX_EXTEND_ROLE parameter in the **onconfig** file to control which users are authorized to register DataBlade modules or external user-defined routines (UDRs).

IFX_FOLDVIEW configuration parameter

Use the IFX_FOLDVIEW configuration parameter to enable or disable view folding. For certain situations where a view is involved in a query, view folding can significantly improve the performance of the query. In these cases, views are folded into a parent query instead of the query results being put into a temporary table.

onconfig.std *value*

0

range of values

0 or Off (default) = Disables view folding

1 or On = Enables view folding

utilities

onmode -wf or **onmode -wm**

The following types of queries can take advantage of view folding:

- Views that contain a UNION ALL clause and the parent query has a regular join, an Informix join, an ANSI join, or an ORDER BY clause

A temporary table is created and view folding is not performed for the following types of queries that perform a UNION ALL operation involving a view:

- The view has one of the following clauses: AGGREGATE, GROUP BY, ORDER BY, UNION, DISTINCT, or OUTER JOIN (either Informix or ANSI type).
- The parent query has a UNION or UNION ALL clause.

Related concepts:

 Enable view folding to improve query performance (Performance Guide)

Related reference:

“onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

LIMITNUMSESSIONS configuration parameter

Use the LIMITNUMSESSIONS configuration parameter to define the maximum number of sessions that you want connected to Informix.

If you specify a maximum number, you can also specify whether you want Informix to print messages to the **online.log** file when the number of sessions approaches the maximum number.

If the LIMITNUMSESSIONS configuration parameter is enabled and sessions are restricted because of this limit, both regular user threads and DBSA user threads connecting to any database count against the limit. However, a DBSA user is allowed to connect to the server even after the limit has been reached.

Distributed queries against a server are also counted against the limit.

The LIMITNUMSESSIONS configuration parameter is not intended to be used as a means to adhere to license agreements.

onconfig.std *value*

Not set in the onconfig.std file

syntax LIMITNUMSESSIONS *maximum_number_of_sessions, print_warning*

separators

Comma

range of values

maximum_number_of_sessions = 0 to 2,097,152 (2*1024*1024). The default is 0.

print_warning = 0 (off) or 1 (on). The default for this optional value is 0.

takes effect

When the database server is shut down and restarted

utilities

onmode -wf or **onmode -wm**

If the print_warning is set to 1, a warning is triggered when the number of sessions is greater than or equal to 95 percent of the maximum_number_of_sessions value. If print_warning is set to zero, or if it is not set, no warning is issued. No new user sessions can be opened after the maximum_number_of_sessions limit is reached.

If the maximum_number_of_sessions value for the LIMITNUMSESSIONS configuration parameter is set to 0, or if it is not set, there is no limit to the number of sessions that can connect to the server.

The following example specifies that you want a maximum of 100 sessions to connect to the server and you want to print a warning message when the number of connected sessions approaches 100.

```
LIMITNUMSESSIONS 100,1
```

The settings in this example cause a warning to be printed when more than 94 sessions are concurrently connected. Only a member of the DBSA group can start a new session when 100 sessions are already connected.

Use **onmode -wf** or **onmode -wm**, or the equivalent SQL administration API ONMODE commands, to dynamically increase or temporarily disable the LIMITNUMSESSIONS setting. Use this configuration parameter to allow administrative utilities to run if the database server is reaching the maximum_number_of_sessions limit.

LISTEN_TIMEOUT Configuration Parameter

onconfig.std *value*
60

Units Seconds

takes effect

When the database server is stopped and restarted

utilities

onmode -wf onmode-wm

refer to IBM Informix Security Guide

LISTEN_TIMEOUT specifies the number of seconds the server waits for a connection. It can be set to a lower number to guard against faulty connection requests that might indicate a Denial of Service attack. See also information about the MAX_INCOMPLETE_CONNECTIONS configuration parameter on page “MAX_INCOMPLETE_CONNECTIONS Configuration Parameter” on page 1-93.

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

Both the LISTEN_TIMEOUT and the MAX_INCOMPLETE_CONNECTIONS configuration parameters can be changed using the **onmode -wf** option or superseded for a session with the **onmode -wm** option. For more information about **onmode**, see “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25.

LOCKS Configuration Parameter

The LOCKS configuration parameter specifies the initial size of the lock table.

The lock table holds an entry for each lock. If the number of locks allocated exceeds the value of the LOCKS configuration parameter, the database server increases the size of the lock table. The lock table can be increased a maximum of 99 times.

onconfig.std *value*

20000

units Number of locks in the internal lock table

range of values

2,000 through 8,000,000 for 32-bit database servers
2,000 through 500,000,000 for 64-bit database servers

takes effect

When the database server is shut down and restarted

utilities

onstat -k (see “**onstat -k** command: Print active lock information” on page 20-192.)

refer to

- The memory and locking chapters in your *IBM Informix Performance Guide*
- The shared memory chapter in the *IBM Informix Administrator's Guide*

The database server increases the size of the lock table by attempting to double the lock table on each increase. However, the amount added during each increase is limited to a maximum value. For 32-bit platforms, a maximum of 100,000 locks can be added during each increase. Therefore, the total maximum locks allowed for 32-bit platforms is 8,000,000 (maximum number of starting locks) + (99 (maximum number of dynamic lock table extensions) × 100,000 (maximum number of locks added per lock table extension)). For 64-bit platforms, a maximum of 1,000,000 locks can be added during each increase. Therefore, the total maximum locks allowed is 500,000,000 (maximum number of starting locks) + (99 (maximum number of dynamic lock table extensions) × 1,000,000 (maximum number of locks added per lock table extension)).

With the initial lock table stored in resident memory and each additional lock stored in virtual memory, locks can become a resource drain if you have a limited amount of shared memory. The amount of storage occupied by a single lock depends on the word size and operating system, and is subject to change. Currently, the amount of storage ranges from approximately 100 to 200 bytes. You can see the amount of storage required to support additional locks by restarting the server with a different value of the LOCKS configuration parameter (without making other changes), and observing the increase in memory used as shown by "onstat -g mem" for the resident pool.

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

LOGBUFF configuration parameter

Use the LOGBUFF configuration parameter to specify the size in kilobytes for the three logical-log buffers in shared memory.

onconfig.std *value*

64

units Kilobytes

range of values

32 kilobytes through (32767 * page size / 1024) kilobytes

takes effect

When the database server is shut down and restarted

utilities

onstat -l buffer field, second section. See “**onstat -l** command: Print physical and logical log information” on page 20-193.

refer to Logical-log buffer, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

Triple buffering permits user threads to write to the active buffer while one of the other buffers is being flushed to disk. If flushing is not complete by the time the active buffer fills, the user thread begins writing to the third buffer.

If you are using RTO_SERVER_RESTART, a LOGBUFF value of 256 kilobytes is recommended. If the LOGBUFF value is less than 256 kilobytes, a warning message displays when you restart the server. Otherwise, set LOGBUFF to 32 kilobytes for standard workloads or 64 kilobytes for heavy workloads. Choose a value for LOGBUFF that is evenly divisible by the page size. If the value of LOGBUFF is not evenly divisible by the page size, the database server rounds down the size to the nearest value that is evenly divisible by the page size.

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

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

To set the system page size, use one of the utilities listed in “System Page Size” on page 1-40.

LOGFILES configuration parameter

Use the LOGFILES configuration parameter to specify the number of logical-log files that the database server creates during disk initialization.

onconfig.std *value*
6

if not present
6

units Number of logical-log files

range of values
3 through 32,767 (integers only)

takes effect

During disk initialization and when you add a new log file. You add a new log with one of the following utilities.

utilities

onparams (see Chapter 16, “The onparams Utility,” on page 16-1)

refer to The following topics in the *IBM Informix Administrator's Guide*:

- Size of logical-log files, in the chapter on the logical log
- Adding or dropping a logical-log file, in the chapter on managing the logical log

To change the number of logical-log files, add or drop logical-log files.

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

LOG_INDEX_BUILDS Configuration Parameter

onconfig.std *value*

None

if not present

0 (disabled)

range of values

0, 1 (0 = disable, 1 = enable)

takes effect

When the database server is stopped and restarted

utilities

onmode -wf onmode -wm

refer to The following topics in the *IBM Informix Administrator's Reference*:

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- Chapter 14, “The onmode utility,” on page 14-1

LOG_INDEX_BUILDS is used to enable or disable index page logging. If LOG_INDEX_BUILDS is enabled, logical log file space consumption will increase, depending on the size of the indexes. This might lead to logical log file backups being required more frequently. Messages are written to the online.log file when index page logging status changes.

Using **onmode -wm** enables or disables index page logging for the current session only, and does not affect the setting in the onconfig file. If the server is stopped and restarted, the setting in the onconfig file determines whether index page logging is enabled. Therefore, enabling index page logging using **onmode -wm** is not recommended when using RS secondary servers; instead, use **onmode -wf** to update the onconfig file, so that index page logging is enabled after restarting the server. Index page logging is a requirement when using RS secondary servers.

LOG_STAGING_DIR Configuration Parameter

Use the LOG_STAGING_DIR configuration parameter to specify the location of log files received from the primary server when configuring delayed application of log files on RS secondary servers.

The directory specified by the LOG_STAGING_DIR configuration parameter is used to store logs sent from the primary server when using the DELAY_APPLY configuration parameter to delay application of log files on an RS secondary server. Delaying the application of log files allows you to recover quickly from erroneous database modifications by restoring the data from the RS secondary server.

The directory specified by the LOG_STAGING_DIR configuration parameter must be secure. The directory must be owned by user **informix**, must belong to group **informix**, and must not have public read, write, or execute permission.

onconfig.std *value*
None

if not present
not defined

range of values (first parameter)
Any valid, secure directory.

takes effect
when the database server is shut down and restarted.

refer to

- “DELAY_APPLY Configuration Parameter” on page 1-53
- “STOP_APPLY Configuration Parameter” on page 1-145
- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- *RS Secondary Server Latency for Disaster Recovery* in the *IBM Informix Administrator’s Guide*

LOGSIZE configuration parameter

Use the LOGSIZE configuration parameter to specify the size that is used when logical-log files are created. The LOGSIZE configuration parameter does not change the size of existing logical-log files. The total logical-log size is LOGSIZE * LOGFILES.

onconfig.std *value*
10000

if not present
10000

units Kilobytes

range of values
Minimum = 200 Maximum = (ROOTSIZE - PHYSFILE - 512 - (63 * ((pagesize)/1024))) / LOGFILES
The *pagesize* value is platform dependent.

takes effect
When the database server is shut down and restarted. The size of log files added after shared memory is initialized reflects the new value, but the size of existing log files does not change.

utilities
onparams See “**onparams -p**: Change physical-log parameters” on page 16-3.

refer to The following topics in the *IBM Informix Administrator’s Guide*:

- Size of the logical log and logging smart large objects, in the chapter on the logical log
- Changes to LOGSIZE or LOGFILES, in the chapter on managing logical logs
- “LTXHWM Configuration Parameter” on page 1-91

To verify the page size that the database server uses on your platform, use one of the utilities listed in “System Page Size” on page 1-40.

You can dynamically change the value of the LOGSIZE configuration parameter by using the **onmode -wm** or **onmode -wf** command.

LOGSIZE for smart large objects

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

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

LOW_MEMORY_MGR configuration parameter

Use the LOW_MEMORY_MGR configuration parameter to enable automatic low memory management, which you can use to change the default behavior of the server when it reaches its memory limit.

onconfig.std *value*
0

default value
0

range of values

1 = Enables automatic low memory management when the database server starts

0 = Disables automatic low memory management

utilities

onmode -wf only (not **onmode -wm**)

takes effect

When the database server is stopped and restarted or when the LOW_MEMORY_MGR value is changed by using the **onmode -wf** utility.

If you enable the LOW_MEMORY_MGR configuration parameter and configure the low memory manager to use a percentage of the SHMTOTAL value for the start and stop thresholds, use caution when changing the value of SHMTOTAL. Changing the value of the SHMTOTAL configuration parameter value can cause the configuration of automatic low memory management to become invalid, forcing the database server to use the default settings.

To enable automatic low memory management, specify:

LOW_MEMORY_MGR 1

Related reference:

“SHMTOTAL configuration parameter” on page 1-130

“onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

“scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)” on page 21-101

“scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)” on page 21-104

LOW_MEMORY_RESERVE configuration parameter

Use the LOW_MEMORY_RESERVE configuration parameter to reserve a specific amount of memory for use when critical activities are needed and the server has limited free memory.

If you enable the new LOW_MEMORY_RESERVE configuration parameter by setting it to a specified value in kilobytes, critical activities, such as rollback activities, can complete even when you receive out-of-memory errors.

onconfig.std *value*

0

default value

0

range of values

0 or 128 to 2147483648, although the maximum value cannot be higher than 20 percent of the value of the SHMVIRTSIZE configuration parameter

utilities

onmode -wf or **onmode -wm**

takes effect

When the database server is stopped and restarted or when the LOW_MEMORY_RESERVE value is changed by using **onmode -wf** or **onmode -wm**.

No matter how the LOW_MEMORY_RESERVE configuration parameter is set, the maximum size of reserved memory is 20 percent of the value of the SHMVIRTSIZE configuration parameter.

For example, specify:

```
LOW_MEMORY_RESERVE 512K
```

You can use the **onstat -g seg** command to view low-memory reserve information. The output includes lines that show the size of reserved memory, the number of times that the server has used the reserved memory, and the maximum memory needed.

Related reference:

“**onstat -g seg** command: Print shared memory segment statistics” on page 20-161

“SHMVIRTSIZE configuration parameter” on page 1-132

“**onmode -wf, -wm**: Dynamically change certain configuration parameters” on page 14-25

LTXEHWM Configuration Parameter

onconfig.std *value*
80

if not present

90 (if DYNAMIC_LOGS is set to 1 or 2) 60 (if DYNAMIC_LOGS is set to 0)

units Percent

range of values

LTXHWM through 100

takes effect

When the database server is shut down and restarted

refer to

- “DYNAMIC_LOGS Configuration Parameter” on page 1-69
- “LTXHWM Configuration Parameter”
- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- Setting high-watermarks for rolling back long transactions, in the chapter on managing logical logs in the *IBM Informix Administrator's Guide*

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

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

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

Tip: To allow users to continue to access the logical logs, even during a long transaction rollback, set *LTXEHWM* to *100*. Set *DYNAMIC_LOGS* to *1* or *2* so that the database server can add a sufficient number of log files to prevent long transactions from hanging and to allow long transactions to roll back.

LTXHWM Configuration Parameter

onconfig.std *value*
70

if not present

80 (if DYNAMIC_LOGS is set to 1 or 2) 50 (if DYNAMIC_LOGS is set to 0)

units Percent

range of values

1 through 100

takes effect

When the database server is shut down and restarted

refer to

- “DYNAMIC_LOGS Configuration Parameter” on page 1-69
- “LTXEHWL Configuration Parameter” on page 1-91
- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- Setting high-watermarks for rolling back long transactions, in the chapter on managing logical logs in the *IBM Informix Administrator's Guide*

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

If DYNAMIC_LOGS is set to 1 or 2, the database server can add a sufficient number of log files to complete the transactions or to prevent rollbacks from hanging when you have long transactions.

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

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

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

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

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

MAX_FILL_DATA_PAGES Configuration Parameter

onconfig.std value

0

units Integer

range of values

0 or 1

takes effect

When the database server is stopped and restarted

refer to Reducing disk space through additional rows per page in tables with variable-length rows in the *IBM Informix Performance Guide*

Set the MAX_FILL_DATA_PAGES value to 1 to allow more rows to be inserted per page in tables that have variable-length rows. This setting can reduce disk space, make more efficient use of the buffer pool, and reduce table scan times.

If MAX_FILL_DATA_PAGES is enabled, the server will add a new row to a recently modified page with existing rows if adding the row leaves at least 10 percent of the page free for future expansion of all the rows in the page. If MAX_FILL_DATA_PAGES is not set, the server will add the row only if there is sufficient room on the page to allow the new row to grow to its maximum length.

A possible disadvantage of enabling MAX_FILL_DATA_PAGES and allowing more variable-length rows per page is that the server might store rows in a different physical order. Also, as the page fills, updates made to the variable-length columns in a row could cause the row to expand so it no longer completely fits on the page. This causes the server to split the row onto two pages, increasing the access time for the row.

To take advantage of this setting, existing tables with variable-length rows must be reloaded or existing pages must be modified, followed by further inserts.

MAX_INCOMPLETE_CONNECTIONS Configuration Parameter

Use the MAX_INCOMPLETE_CONNECTIONS configuration parameter to specify the maximum number of incomplete connections in a session.

onconfig.std *value*
1024

units Number of incomplete connections

takes effect

When the database server is stopped and restarted

utilities

onmode -wf onmode-wm

refer to *IBM Informix Security Guide*

After the number specified in the MAX_INCOMPLETE_CONNECTIONS configuration parameter is reached, an error message is written in the online message log stating that the server might be under a Denial of Service attack. See also information about the LISTEN_TIMEOUT configuration parameter, which specifies the number of seconds the server waits for a connection. “LISTEN_TIMEOUT Configuration Parameter” on page 1-84.

Depending on the machine capability of holding the threads (in number), you can configure MAX_INCOMPLETE_CONNECTIONS to a higher value. Depending on the network traffic, you can also set the LISTEN_TIMEOUT configuration parameter, which specifies the number of seconds the server waits for a connection, to a lower value to reduce the chance that an attack can reach the maximum limit. For more information, see “LISTEN_TIMEOUT Configuration Parameter” on page 1-84.

Both the MAX_INCOMPLETE_CONNECTIONS and the LISTEN_TIMEOUT configuration parameters can be changed using the **onmode -wf** option or

superseded for a session with the **onmode -wm** option. For more information, see “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25.

MAX_PDQPRIORITY Configuration Parameter

onconfig.std *value*

100

if not present

100

range of values

0 through 100

takes effect

On all user sessions

utilities

onmode -D onstat -g mgm (See “**onstat -g mgm** command: Print MGM resource information” on page 20-116.)

refer to

- The chapter on using PDQ, in the *IBM Informix Performance Guide*
- “onmode -D, -M, -Q, -S: Change decision-support parameters” on page 14-11

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

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

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

You can set MAX_PDQPRIORITY to one of the following values.

Value Database Server Action

- | | |
|------------|---|
| 0 | Turns off PDQ. DSS queries use no parallelism. |
| 1 | Fetches data from fragmented tables in parallel (parallel scans) but uses no other form of parallelism. |
| 100 | Uses all available resources for processing queries in parallel. |

number

An integer between 0 and 100. Sets the percentage of the user-requested PDQ resources actually allocated to the query.

MIRROR Configuration Parameter

onconfig.std *value*
0

range of values

0 = disable mirroring 1 = enable mirroring

takes effect

When the database server is shut down and restarted

utilities

onstat -d *flags* field (see “**onstat -d** command: Print chunk information” on page 20-34.)

refer to The following topics in the *IBM Informix Administrator's Guide*:

- Mirroring critical data in the chapter on where is data stored
- Enabling mirroring in the chapter on using mirroring

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

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

MIRROROFFSET Configuration Parameter

onconfig.std *value*
0

units Kilobytes

range of values

Any value greater than or equal to 0

takes effect

When the database server is shut down and restarted

refer to Mirroring the root dbspace during initialization, in the chapter on using mirroring in the *IBM Informix Administrator's Guide*

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

MIRRORPATH Configuration Parameter

onconfig.std *value*

On UNIX: \$INFORMIXDIR/tmp/demo_on.root_mirror On Windows: None

range of values

65 or fewer characters

takes effect

When the database server is shut down and restarted

refer to The following material in the *IBM Informix Administrator's Guide*:

- Mirroring the root dbspace during initialization, in the chapter on using mirroring
- Using links, in the chapter on managing disk space

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

Setting Permissions (UNIX)

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

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

MSG_DATE configuration parameter

Use the MSG_DATE configuration parameter to enable the insertion of a date in MM/DD/YY format at the beginning of each message printed to the online log.

onconfig.std *value*

Not in the onconfig.std template file

range of values

0 = OFF (the default)

1 = ON

takes effect

When the database server is shut down and restarted

utilities

onmode -wf or **onmode -wm**

refer to "onmode -wf, -wm: Dynamically change certain configuration parameters" on page 14-25.

In the following example MSG_DATE is set to 1 (ON).

04/10/10 10:26:06 Value of MSG_DATE has been changed to 1.

04/10/10 10:27:35 Value of MSG_DATE has been changed to 1.

MSGPATH configuration parameter

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

onconfig.std *value*

On UNIX: **\$INFORMIXDIR/tmp/online.log** On Windows: **online.log**

range of values

Pathname

takes effect

When the database server is shut down and restarted

utilities

onstat -m to view the message log (For more information, see “**onstat -l** command: Print physical and logical log information” on page 20-193.)

refer to Message log, in the chapter on overview of database server administration in the *IBM Informix Administrator's Guide*

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

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

You can dynamically change the value of the MSGPATH configuration parameter by using the **onmode -wm** or **onmode -wf** command.

MULTIPROCESSOR Configuration Parameter

onconfig.std *value*
0

if not present
Platform dependent

range of values
0 = No multiprocessor 1 = Multiprocessor available

takes effect
When the database server is shut down and restarted

refer to CPU virtual processors, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*

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

NET_IO_TIMEOUT_ALARM configuration parameter

Use the NET_IO_TIMEOUT_ALARM configuration parameter to control whether to be notified if network write operations have been blocked for 30 minutes or more.

onconfig.std
Not in onconfig.std

default value
0

range of values
One of the following values or a sum of one or more of the following values:

- 0 = Disabled
- 1 = Enabled for Enterprise Replication operations
- 2 = Enabled for distributed queries
- 4 = Enabled for HDR operations

- 8 = Enabled for SMX operations
- 16 = Enabled for other component operations

takes effect

When the database server is shut down and restarted or when the parameter is reset dynamically using the **onmode -wm** or **onmode -wf** commands

Blocked network write operations usually indicate an operating system problem. Use the NET_IO_TIMEOUT_ALARM configuration parameter to enable event alarm 82 for specific types of network traffic.

NETTYPE Configuration Parameter

syntax NETTYPE *protocol,poll_threads,conn_per_thread,VP_class*

onconfig.std values

On UNIX: ipcshm,1,50,CPU

On Windows: None

if not present

protocol:

- On UNIX: The value of **protocol** field from the **sqlhosts** file (with or without the database server prefix of **on**, **ol**, or **dr**)
- On Windows: onsocket

poll_threads: 1

conn_per_thread: 50

VP_class:

- CPU, if the **dbservername** is defined by the DBSERVERNAME configuration parameter.
- NET if the **dbservername** is defined by the DBSERVERALIASES configuration parameter.

separators

Commas

range of values

protocol: Specify as *iiipp* where:

- *iii* is one of the following interface type values:
 - ipc
 - soc
 - tli
- *ppp* is one of the following IPC mechanism or the network protocol values:
 - shm
 - str
 - tcp
 - imc
 - ssl
 - nmp

poll_threads:

- On UNIX: If *VP_class* is NET, a value greater than or equal to 1. If *VP_class* is CPU, 1 through the number of CPU VPs.
- On Windows: Any value greater than or equal to 1.

conn_per_thread: 1 through 32767. For shared memory connection, the value of *conn_per_thread* is the maximum number of connections per thread. For network connections, the value of *conn_per_thread* can be exceeded.

VP_class: CPU = CPU VPs (on UNIX), NET = Network VPs

takes effect

When the database server is shut down and restarted

utilities

onstat -g nsc (see “**onstat -g** monitoring options” on page 20-42.) **onstat -g nss onstat -g nta**

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

- Connecting to IBM data server clients
- Allocating poll threads for an interface/protocol combination
- Setting the size of the DRDA communications buffer
- Displaying DRDA thread and session information
- Network protocol entry, in the chapter on client/server communications
- Multiplexed connections, in the chapter on client/server communications
- Network virtual processors, in the chapter on virtual processors
- Should poll threads run on CPU or network virtual processors, in the chapter on virtual processors
- Monitoring and tuning the number of poll threads and connections, in the *IBM Informix Performance Guide*

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

Configuring a server instance for Secure Sockets Layer (SSL) connections, in the *IBM Informix Security Guide*

The **NETTYPE** parameter provides tuning options for the protocols that **dbservername** entries define in the **sqlhosts** file or registry. Each **dbservername** entry in the **sqlhosts** file or registry is defined on either the **DBSERVERNAME** parameter or the **DBSERVERALIASES** parameter in the **onconfig** file.

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

The **NETTYPE** configuration parameter describes a network connection as follows:

- The protocol (or type of connection)
- The number of poll threads assigned to manage the connection
- The expected number of concurrent connections per poll thread
- The class of virtual processor that will run the poll threads

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

```
NETTYPE ipcshm,3,,CPU
NETTYPE soctcp,8,300,NET
```

The **NETTYPE** parameter for the shared-memory connection (**ipcshm**) specifies three poll threads to run in CPU virtual processors. The number of connections is not specified, so it is set to 50. For **ipcshm**, the number of poll threads correspond to the number of memory segments.

The **NETTYPE** parameter for the sockets connection (**soctcp**) specifies that only 20 simultaneous connections are expected per thread for this protocol and that one poll thread (because the number of poll threads is not specified) will run in a network virtual processor (in this case, NET).

On Windows, if you specify the **soctcp** protocol, only one poll thread is created, and instead, a socket I/O thread (**socketpio**) is created in its own SOC VP for each poll thread specified by the **NETTYPE** parameter. Socket IO threads handle receive operations for all connections using I/O completion ports to receive completion notifications. These threads perform the bulk of the work of servicing network connections on Windows platforms.

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

Number of connections

If only a few connections will be using a protocol concurrently, you might save memory by explicitly setting the estimated number of connections.

For shared memory (**ipcshm**), double the number of connections.

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

Class of virtual processor

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

If you use the VP classes **tli**, **shm**, **str**, or **soc** in the settings for the **VPCLASS** configuration parameter, you must use the class of virtual processor class **NET** for the **NETTYPE** configurator parameter. For more information on the **VPCLASS** configuration parameter, see “**VPCLASS** configuration parameter” on page 1-156.

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

Multiplexed connections

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

```
NETTYPE sqlmux
```

NETTYPE sqlmux does not need to be present in the onconfig file. For more information on enabling multiplexed connections, see the *IBM Informix Administrator's Guide*.

IBM Informix MaxConnect

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

Related reference:

"DBSERVERNAME configuration parameter" on page 1-47

"DBSERVERALIASES configuration parameter" on page 1-45

"NUMFDSERVERS configuration parameter" on page 1-102

"VPCLASS configuration parameter" on page 1-156

NS_CACHE configuration parameter

Use the NS_CACHE configuration parameter to define the maximum retention time for an individual entry in each of these caches: the host name/IP address cache, the service cache, the user cache, and the group cache.

onconfig.std value

host = 900 (seconds)

service = 900 (seconds)

user = 900 (seconds)

group = 900 (seconds)

other values

0 (Disabled. The server always gets information from the operating system.)

Number of seconds

utilities

onmode -wf or **onmode -wm**

takes effect

When the database server is stopped and restarted or when the NS_CACHE value is changed by using **onmode -wf** or **onmode -wm**.

For example, to define the maximum retention time for your host, service, user, and group NS_cache entries as 600 seconds each, specify:

```
NS_CACHE host=600,service=600,user=600,group=600
```

Usage

For looking up and resolving host names (or IP addresses), service names, users (and passwords) or groups, the database server queries the operating system (OS) using appropriate system calls. You can avoid many of these OS lookups by using the Informix name service caching mechanism, which can keep and reuse each retrieved piece of information for a configurable amount of time.

The server can get information from the cache faster than it does when querying the operating system. However, if you disable one or more of these caches by setting the retention time to 0, the database server queries the operating system for the host, service, user, or group information.

You can configure each cache individually by using either the `NS_CACHE` configuration parameter or **onmode -wf** or **onmode -wm**. If you change a particular cache with **onmode -wf** or **onmode -wm**, the server immediately causes all existing entries in the cache to expire.

To disable all caches, set the value of all caches to 0.

Tips:

- Set the `NS_CACHE` configuration parameter if your operating system does not provide its own caching.
- If changes are made to an element that resides in the Informix host name/IP address cache at the operating system level, the service cache, the user cache, or group cache (for example, a new password for a user), the cached element will not be updated immediately. Instead, Informix picks up the change after the existing entry expires.
- When you rely on Informix cached information in any of the four name services, changes made in the operating system (for example, the change of an IP address, a user added to or removed from a group, or a new password) are not immediately effective in the database server. However, you can use an **onmode -wm** or **onmode -wf** command to change `NS_CACHE` information immediately.

NUMFDSERVERS configuration parameter

For network connections on UNIX, use the `NUMFDSERVERS` configuration parameter to specify the maximum number of poll threads to handle network connections migrating between Informix virtual processors (VPs).

Specifying `NUMFDSERVERS` information is useful if Informix has a high rate of new connect and disconnect requests or if you find a high amount of contention between network shared file (NSF) locks. You can use the **onstat -g ath** command to display information about all threads. This information includes a status, such as `mutex wait nsf.lock`, which indicates that you have a significant amount of NSF lock contention.

onconfig.std *value*
4

default value

4 (Only the first 4 poll threads of each **nettype** are involved in managing the connection migrations.)

range of values
1 to 50

The actual number depends on the number of poll threads, which you specify in the NETTYPE configuration parameter.


takes effect

When the database server is shut down and restarted

The specified value of NUMFDSERVERS has no effect on shared-memory (SHM) connections.

If you use the NUMFDSERVERS configuration parameter, also review, and if necessary, change the number of poll threads in the NETTYPE configuration parameter. For example, if you have multiple CPU VPs and poll threads and this results in NSF locking, you can increase NUMFDSERVERS and poll threads to reduce NSF lock contention.

Related concepts:

 Improve connection performance and scalability (Performance Guide)

Related reference:

“NETTYPE Configuration Parameter” on page 1-98

“DBSERVERNAME configuration parameter” on page 1-47

“DBSERVERALIASES configuration parameter” on page 1-45

“**onstat -g ath** command: Print information about all threads” on page 20-54

OFF_RECVRY_THREADS configuration parameter

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

onconfig.std *value*
10

units Number of recovery threads that run in parallel

range of values
Positive integers

takes effect
When the database server is shut down and restarted

refer to

- *IBM Informix Backup and Restore Guide*
- *IBM Informix Performance Guide*

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

ON_RECVRY_THREADS Configuration Parameter

onconfig.std *value*
1

units Number of recovery threads that run in parallel

range of values

Positive integers

takes effect

When the database server is shut down and restarted

refer to

- *IBM Informix Backup and Restore Guide*
- *IBM Informix Performance Guide*

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

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

To improve the performance of warm restores, increase the number of fast-recovery threads with the ON_RECOVERY_THREADS parameter.

ONDBSPACEDOWN configuration parameter

Use the ONDBSPACEDOWN configuration parameter to define the action that the database server takes when any disabling event occurs on a primary chunk within a noncritical dbspace.

onconfig.std *value*

2

range of values

0, 1, 2

refer to Monitoring the database server for disabling I/O errors, in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

The following values are valid for this parameter:

Value	Description
-------	-------------

- | | |
|---|---|
| 0 | The database server marks the dbspace as offline and continues. |
| 1 | The database server aborts. |
| 2 | The database server writes the status of the chunk to the logs and waits for user input. If you set this option, but you want the database server to mark a disabled dbspace as down and continue processing, use onmode -O to override this ONDBSPACEDOWN setting. See “onmode -O: Override ONDBSPACEDOWN WAIT mode” on page 14-19. |

You can dynamically change the value of the ONDBSPACEDOWN configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Database Server Behavior When ONDBSPACEDOWN Does Not Apply

The database server will not come online if a chunk within any **critical** dbspace (for example, rootdbs or logsdbs) is missing.

The value of ONDBSPACEDOWN has no effect on temporary dbspaces. For temporary dbspaces, the database server continues processing regardless of the ONDBSPACEDOWN setting. If a temporary dbspace requires fixing, you should drop and recreate it.

For a non-primary chunk within a noncritical dbspace, the behavior of the database server depends on the transaction status of the chunk when the disabling event occurs:

- **No transaction:** If no transactions are detected against that chunk, the chunk is individually marked as down. In this case, subsequent attempts to write to that chunk fail, rolling back the associated transaction. You can safely put the chunk back and then use the **onspaces -s** utility to mark the chunk as back online.
- **Transaction detected:** If there are transactions to roll forward or back, then the database server aborts with an appropriate fast recovery error. In this case, you should put the chunk back and restart the database server.

ONLIDX_MAXMEM Configuration Parameter

onconfig.std value

5120

units Kilobytes

range of values

16 through 4294967295

takes effect

When the database server is shut down and restarted

utilities

onmode -wf onmode-wm

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

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

The ONLIDX_MAXMEM configuration parameter can be changed using the **onmode -wf** option or superseded for a session with the **onmode -wm** option. For more information about **onmode**, see “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25.

OPCACHEMAX Configuration Parameter (UNIX)

onconfig.std *value*

0

if not present

128

units Kilobytes

range of values

0 through (4 * 1024 * 1024)

takes effect

When the Optical Subsystem needs more memory

utilities

onstat -O (For more information, see “**onstat -o** command: Output shared memory contents to a file” on page 20-198.)

refer to

- *IBM Informix Optical Subsystem Guide*
- **INFORMIXOPCACHE** environment variable, in the *IBM Informix Guide to SQL: Reference*

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

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

Related reference:

 **INFORMIXOPCACHE** environment variable (SQL Reference)

OPTCOMPIND configuration parameter

Use the OPTCOMPIND to specify information that helps the optimizer choose an appropriate query plan for your application.

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

onconfig.std *value*

2

range of values

0 = When appropriate indexes exist for each ordered pair of tables, the optimizer chooses index scans (nested-loop joins), without consideration of the cost, over table scans (hash joins). This value ensures compatibility with previous versions of the database server.

1 = The optimizer uses costs to determine an execution path if the isolation level is not Repeatable Read. Otherwise, the optimizer chooses index scans (it behaves as it does for the value 0). This setting is recommended for optimal performance.

2 = The optimizer uses cost to determine an execution path for any isolation level. Index scans are not given preference over table scans; the optimizer bases its decision purely on cost. This value is the default if the variable is not set.

refer to

- Your *IBM Informix Performance Guide*
- **OPTCOMPIND** environment variable, in the *IBM Informix Guide to SQL: Reference*
- SET ENVIRONMENT OPTCOMPIND statement to dynamically change the value of the OPTCOMPIND configuration parameter for a session, in the *IBM Informix Guide to SQL: Syntax*

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

You can dynamically change the value of the OPTCOMPIND configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related concepts:

 OPTCOMPIND Environment Option (SQL Syntax)

Related reference:

 OPTCOMPIND environment variable (SQL Reference)

OPT_GOAL Configuration Parameter

onconfig.std *value*
-1

range of values
0 or -1

takes effect
When the database server is shut down and restarted

refer to The following manuals:

- ALL_ROWS and FIRST_ROWS directives and on the SET OPTIMIZATION statement, in the *IBM Informix Guide to SQL: Syntax*
- **OPT_GOAL** environment variable, in the *IBM Informix Guide to SQL: Reference*
- Performance issues associated with setting an optimization goal, in the *IBM Informix Performance Guide*

The OPT_GOAL parameter enables you to specify one of the following optimization goals for queries:

- Optimize for FIRST ROWS
- Optimize for ALL ROWS

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

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

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

You can specify the optimization goal in one of four ways:

- By query (SELECT statement)
Use the ALL_ROWS and FIRST_ROWS directives.
- By session
Use the SET OPTIMIZATION statement.
- By environment
Set the **OPT_GOAL** environment variable.
- By database server
Set the OPT_GOAL configuration parameter.

The list above lists the mechanisms for setting this goal in descending order of precedence. To determine the optimization goal, the database server examines the settings in the order above. The first setting encountered determines the optimization goal. For example, if a query includes the ALL_ROWS directive but the OPT_GOAL configuration parameter is set to FIRST_ROWS, the database server optimizes for ALL_ROWS, as the query specifies.

Related reference:

 [OPT_GOAL environment variable \(UNIX\) \(SQL Reference\)](#)

PC_HASHSIZE Configuration Parameter

onconfig.std *value*
31

range of values

Any positive integer, a prime number is recommended.

takes effect

When the database server is shut down and restarted

refer to Your IBM Informix Performance Guide

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

PC_HASHSIZE applies to UDR cache only. For information on configuration parameters for other types of cache, see the “DS_POOLSIZE Configuration Parameter” on page 1-64 and the “DS_HASHSIZE Configuration Parameter” on page 1-61.

PC_POOLSIZE Configuration Parameter

onconfig.std *value*
127

default value
127

range of values
Any positive value from 127 to x , where x is dependent upon the shared memory configuration and available shared memory for the server instance.

takes effect
When the database server is shut down and restarted

refer to Your IBM Informix Performance Guide

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

For information on configuration parameters for other types of cache, see the “DS_POOLSIZE Configuration Parameter” on page 1-64 and the “DS_HASHSIZE Configuration Parameter” on page 1-61.

PHYSBUFF Configuration Parameter

onconfig.std *value*
128

units Kilobytes

range of values
4 kilobytes through $(32767 * \text{page size} / 1024)$ kilobytes.

takes effect
When the database server is shut down and restarted

utilities
onstat -l *buffer* field, first section (For more information, see “**onstat -l** command: Print physical and logical log information” on page 20-193.)

refer to Physical-log buffer, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

PHYSBUFF specifies the size in kilobytes of the two physical-log buffers in shared memory. Double buffering permits user threads to write to the active physical-log buffer while the other buffer is being flushed to the physical log on disk. The value of the PHYSBUFF parameter determines how frequently the database server needs to flush the physical-log buffer to the physical-log file. If RTO_SERVER_RESTART is enabled, use the 512 kilobyte default value for PHYSBUFF; if the PHYSBUFF value is less than 512 kilobytes, a warning message displays when you restart the server.

A write to the physical-log buffer is exactly one page in length. Choose a value for PHYSBUFF that is evenly divisible by the page size. If the value of PHYSBUFF is

not evenly divisible by the page size, the database server rounds down the size to the nearest value that is evenly divisible by the page size.

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

The system page size is platform-dependent on Informix. To obtain the system page size, use the commands listed in the table in “System Page Size” on page 1-40.

PHYSFILE Configuration Parameter

Use the PHYSFILE configuration parameter to specify the size or location of the physical log file when you first initialize the disk space and bring the database server online.

onconfig.std *value*
50000

if not present
200

units Kilobytes

range of values
200 or more

takes effect

After the disk space has been initialized using the **oninit -i** command, the server does not dynamically implement changes that you make to the PHYSFILE configuration parameter. You can force the change, but that requires you to reinitialize the database server. Instead, if the database server is online, use the **onparams** utility to change the size or location of physical log file. You do not need to restart the server for the changes to take effect.

utilities

onparams -p

refer to The following topics in the *IBM Informix Administrator's Guide*:

- Sizing the physical log, in the chapter on the physical log
- Changing the physical-log location and size, in the chapter on managing the physical log

When the RTO_SERVER_RESTART configuration parameter is enabled, ensure that the value for the PHYSFILE configuration parameter is equal to at least 110% of the buffer pool size. A warning message prints to the message log when:

- The value for the PHYSFILE configuration parameter is changed to less than 110% of all of the buffer pools
- The server is restarted
- A new buffer pool is added

PLOG_OVERFLOW_PATH Configuration Parameter

onconfig.std *value*
On UNIX: \$INFORMIXDIR/tmp On Windows: None

if not present

\$INFORMIXDIR/tmp

takes effect

When the database server is brought up (shared memory is initialized)

refer to Your IBM Informix Administrator's Guide

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

PN_STAGEBLOB_THRESHOLD configuration parameter

Use the PN_STAGEBLOB_THRESHOLD configuration parameter to reserve space for BYTE and TEXT data in round-robin fragments.

Set this configuration parameter to the typical or average size of the BYTE or TEXT data that is stored in the table.

Restriction: The PN_STAGEBLOB_THRESHOLD configuration parameter has no effect if the number of extents has reached the maximum extents allowed or if the dbspace is full.

When a table reaches the maximum number of pages for a fragment, more pages can be added to the table by adding a new fragment. However, if a table contains BYTE or TEXT columns and that table is fragmented by the round-robin distribution scheme, adding a new fragment does not automatically enable new rows to be inserted into the new fragment.

For example, if one of the fragments in the table reaches the maximum number of pages, adding a new fragment does not extend the table to store more rows. Because BYTE and TEXT data tend to be large in size, the data is *staged* in one of the fragments before being distributed evenly in all of the fragments. The staging fragment must have sufficient space to store the BYTE or TEXT data. Use the PN_STAGEBLOB_THRESHOLD configuration parameter so that the database server can stage the BYTE or TEXT data temporarily in a staging fragment until the INSERT operation is completed and the data is permanently stored in the table.

During a UPDATE operation if the fragment does not have the space that is specified in PN_STAGEBLOB_THRESHOLD configuration parameter the table row that is impacted by the updated is moved into another fragment.

onconfig.std *value*

None

if not present

0

syntax

PN_STAGEBLOB_THRESHOLD *value*

range of values

0 to 1000000

units

Kilobytes

takes effect

When the database server is shut down and restarted

Related concepts:

 Fragmentation by ROUND ROBIN (Data Warehouse Guide)

PRELOAD_DLL_FILE Configuration Parameter

The PRELOAD_DLL_FILE configuration parameter specifies the path name for a shared library file that is preloaded when the database server is started.

You must specify a separate occurrence of this parameter for each library file that you want to preload. Use this parameter to preload the shared library files for the user-defined routines that are created in the C programming language (C UDRs).

onconfig.std *value*

None

takes effect

When the database server is started.

value Full path name for the shared library file. The following are some examples:

- /finance/jeffzhang/mylib.udr
- home/informix/extend/blade.so
- /deptxyz/udrs/geodetic.bld
- \work4\lauragupta\userfuncdir\cudrs.ddl

utilities

onstat -g dll

Restriction: The PRELOAD_DLL_FILE configuration parameter is not supported by the **onmode -wm** command.

Restriction: A preloaded C UDR shared library remains active until the database instance is brought offline.

refer to onstat -g dll command. If the P flag is set, the shared library is preloaded on the server.

QSTATS Configuration Parameter

onconfig.std *value*

0

range of values

0 = Disable queue statistics 1 = Enable queue statistics

takes effect

When the database server is shut down and restarted

utilities

onstat -g qst

refer to

- “**onstat -g qst** command: Print wait options for mutex and condition queues” on page 20-141

QSTATS specifies the ability of **onstat -g qst** to print queue statistics.

RA_PAGES configuration parameter

If the AUTO_READAHEAD configuration parameter is set to 0 in the ONCONFIG file, the database server ignores values set in the RA_PAGES configuration parameter.

When the AUTO_READAHEAD configuration parameter is enabled and read ahead occurs automatically, the server ignores information specified in the RA_PAGES configuration parameter.

onconfig.std *value*
64

if not present

4 if MULTIPROCESSOR is 0

8 if MULTIPROCESSOR is 1

units Number of data pages

range of values

4 to 4096. However, this value is also limited to the size of the bufferpool. For example, if the bufferpool has only 1000 pages, then the maximum value of the RA_PAGES configuration parameter is 1000.


takes effect

When the database server is shut down and restarted

The read-ahead threshold, which is the number of unprocessed data pages in memory that signals the database server to perform the next read-ahead, is one-half of the number of pages that is set for the RA_PAGES configuration parameter. If the AUTO_READAHEAD configuration parameter is set to 0, try setting the RA_PAGES configuration parameter to 64.


You can dynamically change the value of the RA_PAGES configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related concepts:

 Sequential scans (Performance Guide)

Related reference:

“AUTO_READAHEAD configuration parameter” on page 1-31

 Configure the database server to read ahead (Administrator's Guide)

REMOTE_SERVER_CFG configuration parameter

Use the REMOTE_SERVER_CFG configuration parameter to specify the name of a file that lists the remote hosts that are trusted by the computer on which the database server resides.

The file specified by the REMOTE_SERVER_CFG configuration parameter must be located in \$INFORMIXDIR/etc. If the configuration parameter is set then file specified is used instead of the /etc/hosts.equiv file.

The format of the file specified by the REMOTE_SERVER_CFG configuration parameter is the same as the format of the hosts.equiv file. If all the servers should trust the same set of remote servers, then the file can specify either file names or symbolic links; alternatively, the same name can be used for REMOTE_SERVER_CFG in each remote server's ONCONFIG file.

Consider using the following naming convention for the file specified by the REMOTE_SERVER_CFG configuration parameter:

`$INFORMIXDIR/etc/servers.server_name`

onconfig.std *value*
Not specified

range of values
File name. The path is not specified in onconfig and is assumed to be `$INFORMIXDIR/etc`.

takes effect
If the parameter was changed in the onconfig file, or if the **onmode -wf** command was used, the parameter takes effect when the database server is shut down and restarted. Alternatively, you can dynamically change the value of the REMOTE_SERVER_CFG configuration parameter by using the **onmode -wm** command.

utilities
onmode -wf or **onmode -wm**

refer to “onmode -wf, -wm: Dynamically change certain configuration parameters”
on page 14-25

REMOTE_USERS_CFG configuration parameter

Use the REMOTE_USERS_CFG configuration parameter to specify the name of a file that lists names of trusted users that exist on remote hosts.

The file specified by the REMOTE_USERS configuration parameter must be located in `$INFORMIXDIR/etc`. If the configuration parameter is set then the file specified is used instead of the `~/ .rhosts` file.

The format of the file specified by the REMOTE_USERS_CFG configuration parameter is the same as the format of the `~/ .rhosts` file.

Consider using the following naming convention for the file specified by the REMOTE_USERS_CFG configuration parameter:

`$INFORMIXDIR/etc/users.server_name`

onconfig.std *value*
Not specified

range of values
File name. The path is not specified in onconfig and is assumed to be `/INFORMIX/etc`.

takes effect
When the database server is shut down and restarted

utilities
onmode -wf or **onmode -wm**

refer to “onmode -wf, -wm: Dynamically change certain configuration parameters”
on page 14-25

RESIDENT configuration parameter

Use the RESIDENT configuration parameter to specify whether resident and virtual segments of shared memory remain resident in operating-system physical memory.

onconfig.std *value*
0

range of values

-1 to 99

0 = off

1 = lock the resident segment only

-1 = lock all resident and virtual segments

n = lock the resident segment and the next $n - 1$ virtual segments. For example, if you specify 99 as the value, the resident segment is locked and the next 98 virtual segments are locked.

Certain platforms have different values. For information, see your machine notes.

if not present
0

takes effect

When the database server is shut down and restarted

utilities

onmode -r (see “onmode -n, -r: Change shared-memory residency” on page 14-18)

refer to The following topics in the *IBM Informix Administrator's Guide* for a discussion of residency:

- Resident portion of shared memory, in the shared-memory chapter
- Setting database server shared-memory configuration parameters, in the chapter on managing shared memory

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

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

On AIX, Solaris, or Linux systems that support large pages of memory, the DBSA can use operating system commands to configure a pool of large pages.

Informix can store non-message virtual memory segments on these large pages if you take the following steps:

- Enable large page sizes by setting the **IFX_LARGE_PAGES** environment variable.
- For virtual memory segments that you intend to store on large pages, set the RESIDENT parameter to lock those segments in physical memory, so that they cannot be swapped to disk

Storing virtual memory segments on large pages can offer significant performance benefits in large memory configurations. For more information about enabling Informix to use shared memory segments that are resident on large pages, see the description of **IFX_LARGE_PAGES** in the chapter about environment variables in your *IBM Informix Guide to SQL: Reference*.

Related reference:

 **IFX_LARGE_PAGES** environment variable (SQL Reference)

RESTARTABLE_RESTORE Configuration Parameter

onconfig.std *value*
ON

if not present
ON

range of values
OFF = restartable restore is disabled ON = restartable restore is enabled

takes effect
When the database server is shut down and restarted

refer to *IBM Informix Backup and Restore Guide*

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

Increase the size of your physical log if you plan to use restartable restore. For more information, see “**PHYSFILE** Configuration Parameter” on page 1-110. Although a restartable restore slows down the logical restore if many logs need to be restored, you save a lot of time from not having to repeat the entire restore.

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

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

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

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

RESTORE_POINT_DIR configuration parameter

Use the RESTORE_POINT_DIR configuration parameter to change the path name of the directory where restore point files will be placed during a failed upgrade to a new version of the server. Informix will store restore point files in a subdirectory of the specified directory, with the server number as the subdirectory name, only if the CONVERSION_GUARD configuration parameter is enabled.

onconfig.std *value*
\$INFORMIXDIR/tmp

range of values
complete path name for a directory

takes effect
When the database server is stopped and restarted

You can change the directory, for example, if you think that the \$INFORMIXDIR/tmp directory does not have enough space for restore point data. If you want to change the directory, you must change it before you initiate an upgrade to a new version of the server. You cannot change the directory during an upgrade.

The directory specified in the RESTORE_POINT_DIR configuration parameter must be empty when an upgrade begins. If the directory contains any restore point files from a previous upgrade, you must remove the files before a new upgrade begins a new restore point.

For information about the **onrestorept** utility, which you can use to undo changes made during a failed upgrade, see the *IBM Informix Migration Guide*.

ROOTNAME Configuration Parameter

onconfig.std *value*
rootdbs

units A dbspace

range of values
Up to 128 bytes. ROOTNAME must begin with a letter or underscore and must contain only letters, numbers, underscores, or \$ characters.

takes effect
When disk is initialized (destroys all data)

refer to Allocating disk space, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*

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

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

ROOTOFFSET Configuration Parameter

onconfig.std *value*
0

units Kilobytes

range of values

Any value greater than or equal to 0

takes effect

When disk is initialized (destroys all data)

refer to Allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*

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

UNIX Only:

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

ROOTPATH Configuration Parameter

Use the ROOTPATH configuration parameter to specify the full pathname, including the device or filename, of the initial chunk of the root dbspace. The ROOTPATH configuration parameter is stored in the reserved pages as a chunk name.

onconfig.std *value*

On UNIX: **\$INFORMIXDIR/tmp/demo_on.rootdbs**

On Windows: None

range of values

Pathname

takes effect

When disk is initialized (destroys all data)

refer to The following material in the chapter on managing disk space in the *IBM Informix Administrator's Guide*

- Allocating disk space
- Creating links for raw devices

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

UNIX Only:

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

ROOTSIZE configuration parameter

Use the ROOTSIZE configuration parameter to specify the size in kilobytes of the initial chunk of the root dbspace. The size that you select depends on your immediate plans for your database server.

onconfig.std *value*

200000

if not present

0

units Kilobytes

range of values

50,000 through maximum capacity of the storage device

takes effect

When disk is initialized (destroys all data)

The database server uses the value of the ROOTSIZE configuration parameter only during a complete disk initialization. Changing the ROOTSIZE value after the initial chunk of the root dbspace has been created will have no effect.

Related concepts:

 Size of the root dbspace (Administrator's Guide)

RSS_FLOW_CONTROL Configuration Parameter

Specifies the boundaries within which flow control is enabled in a high-availability cluster that contains at least one RS secondary server.

onconfig.std *value*

0

range of values

Table 1-56. Valid values for the RSS_FLOW_CONTROL configuration parameter.

Value	Description
0	Flow control is activated when the difference between the current log position and the most recent acknowledged log exceeds eight times the size of the log buffer.
-1	Flow control is disabled. Disabling flow control might lead to wrapping of the log files and the loss of data.
<i>start_value, end_value</i>	The <i>start_value</i> and <i>end_value</i> parameters determine the amount of lag between the current log position and the last acknowledged log page. The values for <i>start_value</i> and <i>end_value</i> have a scaling factor of Kilobytes (K), Megabytes (M), or Gigabytes (G). For example, setting the RSS_FLOW_CONTROL configuration parameter to 128M,100M starts RSS flow control when the lag between the logs is 128 megabyte and stops flow control when the log lag has dropped to 100 megabytes. The <i>start_value</i> must be greater than the <i>end_value</i> .

takes effect

When the database server is stopped and restarted or when the RSS_FLOW_CONTROL value is changed by using **onmode -wm** or **onmode -wf**.

utilities

onmode -wf or **onmode -wm**

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

Flow control provides a way to limit log activity on the primary server so that RS secondary servers in the cluster do not fall too far behind on processing transactions. When flow control is enabled, and when the difference in log size between the current log position and the last acknowledged log page exceeds the value set by *start_value*, then log activity on the primary server becomes restricted. Users connected to the primary server may experience slower response time when flow control is active.

The `RSS_FLOW_CONTROL` configuration parameter allows you to configure when flow control is started or stopped. You set the `RSS_FLOW_CONTROL` configuration parameter on the primary server only. All RS secondary servers in the cluster are affected by the `RSS_FLOW_CONTROL` configuration parameter.

Flow control is started when the lag between the logs is greater than the *start_value* parameter and stops flow control when the log lag has dropped to the value set by the *stop_value* parameter.

Enabling flow control ensures that logs on RS secondary servers remain current if the RS secondary servers are situated on a busy or intermittent network. Logs are always sent to the RS secondary server in the order in which they were received.

RTO_SERVER_RESTART configuration parameter

Use the `RTO_SERVER_RESTART` configuration parameter to specify recovery time objective (RTO) standards for the amount of time, in seconds, that Informix has to recover from a problem after you restart the server and bring it into online or quiescent mode.

onconfig.std *value*
0 (disabled)

units seconds

range of values
0 = disabled
60 to 1800

takes effect
When the database server is stopped and restarted or when the `RTO_SERVER_RESTART` value is changed by using **onmode -wm** or **onmode -wf**.

utilities
oncheck -pr
onmode -wf or **onmode -wm**
onparams

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- Nonblocking checkpoints information in *IBM Informix Administrator's Guide*
- Effects of configuration on I/O activity in *IBM Informix Enterprise Replication guide*

S6_USE_REMOTE_SERVER_CFG configuration parameter

Used with the REMOTE_SERVER_CFG configuration parameter, the S6_USE_REMOTE_SERVER_CFG configuration parameter specifies the file used to authenticate secure server connections in an untrusted network environment.

The REMOTE_SERVER_CFG configuration parameter is used to specify the name of a file that lists the remote server hosts that are trusted by the computer on which the database server resides. If one or more of the servers specified in the REMOTE_SERVER_CFG file is configured using the connection security option (s=6) in the sqlhosts file, then setting S6_USE_REMOTE_SERVER_CFG to 1 specifies that the servers in the REMOTE_SERVER_CFG are trusted connections. If S6_USE_REMOTE_SERVER_CFG is set to 0, then the \$INFORMIXDIR/etc/host.equiv file is used to authenticate servers connecting through a secure port (as defined by setting s=6 in the **sqlhost** file).

onconfig.std *value*
None

if not present
0 (off)

range of values

- 0 = The \$INFORMIXDIR/etc/host.equiv file is used to authenticate servers connecting through a secure port as defined by setting s=6 in the sqlhost information.
- 1 = The servers listed in the file specified by the REMOTE_SERVER_CFG configuration parameter are trusted connections.

takes effect

If the parameter is changed in the onconfig file, or if the **onmode -wf** command is used, the parameter takes effect when the database server is shut down and restarted. Alternatively, you can dynamically change the value of the S6_USE_REMOTE_SERVER_CFG configuration parameter by using the **onmode -wm** command.

utilities

onmode -wf or **onmode -wm**

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

SBSPACENAME Configuration Parameter

onconfig.std *value*
None

if not present
0

range of values

Up to 128 bytes. SBSPACENAME must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or \$ characters.

takes effect

When shared memory is reinitialized

utilities

onspaces -c -S

refer to

- Using **onspaces** to “onspaces -c -S: Create an sbspace” on page 19-11
- “SBSPACETEMP Configuration Parameter” on page 1-123
- “SYSSBSPACENAME Configuration Parameter” on page 1-148
- “Sbspace Structure” on page 4-24
- What is an sbspace, in the chapter on data storage in the *IBM Informix Administrator's Guide*
- Altering sbspace characteristics, in the chapter on managing data on disk in the *IBM Informix Administrator's Guide*
- Assigning a smart large object to an sbspace, in the section on the CREATE TABLE and ALTER TABLE statements, in the *IBM Informix Guide to SQL: Syntax*
- Creating an sbspace for Enterprise Replication usage, in the IBM Informix Enterprise Replication guide
- Using multirepresentational data, in the *IBM Informix DataBlade API Programmer's Guide*

SBSPACENAME specifies the name of the default sbspace. If your database tables include smart-large-object columns that do not explicitly specify a storage space, that data is stored in the sbspace that SBSPACENAME specifies. The default sbspace is also used by the built-in encryption and decryption functions to store BLOB or CLOB values. If DECRYPT_BINARY or an encryption function cannot find an sbspace in which to store a BLOB or CLOB argument or returned value, the function fails with the following error message:

Fatal error in server row processing - SQL error -9810 ISAM error -12053

If you see this error message after you invoke an encryption or decryption function that has a CLOB or BLOB argument, configure a default sbspace using the SBSPACENAME configuration parameter, and then repeat the function call.

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

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

JAVA Language Support:

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

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

Automatic creation of the default sbspace

Under certain circumstances, a default sbspace is created even if the SBSPACENAME configuration parameter is not set:

- If you create a **bts** index and do not explicitly specify an sbspace name
- If you create a table with a spatial data type column and do not explicitly specify an sbspace name

The default sbspace is created in the root dbspace for the database server with a size of 10 000 KB. You must manually increase the size of the default sbspace when it fills.

SBSPACETEMP Configuration Parameter

onconfig.std *value*

None

if not present

Temporary smart large objects are stored in a standard sbspace.

range of values

Up to 128 bytes. SBSPACETEMP must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or \$ characters.

takes effect

When shared memory is reinitialized

utilities

onspaces

refer to

- “onspaces -c -S: Create an sbspace” on page 19-11
- “SBSPACENAME Configuration Parameter” on page 1-121
- Temporary sbspaces, in the chapter on data storage in the *IBM Informix Administrator's Guide*
- Creating a temporary sbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*
- Using temporary smart large objects, in the *IBM Informix DataBlade API Programmer's Guide*

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

SDS_ENABLE Configuration Parameter

onconfig.std *value*

None

if not present

0

range of values

0, 1 0 = disable, 1 = enable

takes effect

When database server is shut down and restarted.

refer to

- *Using Shared Disk Secondary Servers in the IBM Informix Administrator's Guide.*

SDS_ENABLE enables SD secondary server functionality. You must set SDS_ENABLE to 1 (enable) on the SD secondary server to enable SD secondary server functionality.

SDS_ENABLE is set to 1 (enabled) automatically when you run the following command:

```
onmode -d set SDS primary
```

SDS_ENABLE is set to 0 (disabled) when you run the following command:

```
onmode -d clear SDS primary
```

To prevent data corruption, you cannot use the **oninit -i** or **oninit -iy** command to initialize disk space on a server if SDS_ENABLE is set to 1 (enabled). To initialize an SD secondary server, initialize only the shared memory by using **oninit** with no parameters. To initialize a primary server to which one or more SD secondary servers are attached, and whose disk has never been initialized, set SDS_ENABLE to 0 and initialize the server memory and disk using **oninit -i**. To initialize a primary server to which SD secondary servers are attached, and whose disk is already initialized, set SDS_ENABLE to 1 and initialize shared memory only using **oninit** with no parameters. See Chapter 12, "The oninit utility," on page 12-1.

SDS_LOGCHECK configuration parameter

Use the SDS_LOGCHECK configuration parameter to determine whether the primary server is generating log activity and to allow or prevent failover of the primary server.

For example, if the SDS_LOGCHECK configuration parameter is set to 10, and the primary server fails, the SD secondary server waits up to 10 seconds to either detect that the primary server is generating log records (in which case failover is prevented), or the SD secondary server detects that the primary is not generating log records and failover occurs.

onconfig.std *value*

0

if not present

0

units Seconds

range of values

0 = Do not detect log activity; allow failover.

n = Wait up to *n* seconds. If log activity is detected from the primary server, failover is prevented; otherwise, failover is allowed.

takes effect

When shared disk functionality is enabled on the primary server

refer to

- *Using shared disk secondary servers in the IBM Informix Administrator's Guide*

The SDS_LOGCHECK configuration parameter specifies the amount of time in seconds that the SD secondary server waits to detect whether the primary server is generating log activity. If there is no log activity generated by the primary server in the specified amount of time, failover processing is allowed to start. If log activity from the primary server is detected, failover processing is not started.

Set the SDS_LOGCHECK configuration parameter to a value greater than zero if you do not have I/O fencing configured and your system consists of a primary server and one or more SD secondary servers.

If your system has I/O fencing configured, and if an SD secondary server becomes a primary server, the I/O fencing script must prevent the failed primary server from updating any of the shared disks. If the system does not have I/O fencing configured, the SDS_LOGCHECK configuration parameter prevents the occurrence of multiple primary servers by not failing over to the SD secondary server if the original primary server is generating log records.

SDS_PAGING Configuration Parameter

The SDS_PAGING configuration parameter specifies the location of two files that serve as buffer paging files.

onconfig.std *value*
None

takes effect

When SD secondary server is started

refer to

- *Using Shared Disk Secondary Servers in the IBM Informix Administrator's Guide.*

Set the SDS_PAGING configuration parameter to the paths of two buffer paging files separated by a comma. The SDS_PAGING configuration parameter must be set to a valid value to ensure that the SD secondary server starts. Because the paging files grow dynamically as needed, you should allocate enough disk space to store two times the size of the value specified by the PHYSFILE configuration parameter (see "PHYSFILE Configuration Parameter" on page 1-110)

SDS_TEMPDBS Configuration Parameter

Use the SDS_TEMPDBS configuration parameter to specify information that the shared disk (SD) secondary server uses to dynamically create temporary dbspaces. This configuration parameter can be specified only on the SD secondary server.

The temporary dbspaces are created (or initialized if the dbspaces existed previously) when the SD secondary server starts. The temporary dbspaces are used for creating temporary tables. There must be at least one occurrence of the SDS_TEMPDBS configuration parameter in the ONCONFIG file of the SD secondary server for the SD secondary server to start. You can specify up to 16 SD secondary dbspaces in the ONCONFIG file by using multiple occurrences of the SDS_TEMPDBS configuration parameter.

The format of the SDS_TEMPDBS configuration parameter is:

`SDS_TEMPDBSdbspace,dbspath,pagesize,offset,size`

dbasename

The *dbasename* value must be unique. In addition, the *dbasename* must be unique among all existing dbspaces, blobspaces, and sbspaces, including those (possibly disabled) temporary spaces that are inherited from a primary server. If you have multiple SD secondary servers, the *dbasename* value should be unique for each server and not shared with any other SD secondary server or the primary server.

dbspath

The path that corresponds to the *dbasename*.

pagesize

A numeric value specified in kilobytes.

offset

A numeric value. You can specify a single character modifier with the value to designate the units:

- K or k for kilobytes
- M or m for megabytes
- G or g for gigabytes
- T or t for terabytes

The default unit is kilobytes.

size

A numeric value. The value cannot be zero (0). You can specify a single character modifier with the value to designate the units:

- K or k for kilobytes
- M or m for megabytes
- G or g for gigabytes
- T or t for terabytes

The default unit is kilobytes.

onconfig.std *value*

None

if not present

An error message displayed and the SD secondary server will not start.

range of values

0 on the primary database server, 1 to 16 dbspace entries on an SD secondary server.

takes effect

When the SD secondary server is started.

refer to

- *Using Shared Disk Secondary Servers* in the *IBM Informix Dynamic Server Administrator's Guide*.

Specifying multiple occurrences of the SDS_TEMPDBS configuration parameter

For each occurrence of the SDS_TEMPDBS configuration parameter in the ONCONFIG file:

- The combination of *dbspath*, *pagesize*, *offset*, and *size* must not cause any overlap with existing chunks or among SDS_TEMPDBS spaces.
- The *pagesize* value for each temporary SDS dbspaces must be the same value.

SDS_TIMEOUT Configuration Parameter

onconfig.std *value*

20

if not present

10

range of values

2 to 2147483647

units seconds

takes effect

When shared disk functionality is enabled on the primary server

refer to

- The **onmode -wf** command
- *Using Shared Disk Secondary Servers* in the *IBM Informix Administrator's Guide*.

SDS_TIMEOUT specifies the amount of time in seconds that the primary server will wait for a log position acknowledgment to be sent from the SD secondary server. If there is no log position acknowledgment received from the SD secondary server in the specified amount of time, the primary server will be disconnected from the SD secondary server and continue. After waiting for the SDS_TIMEOUT number of seconds, the primary server will start removing SD secondary servers if page flushing has timed out while waiting for an SD secondary server.

SECURITY_LOCALCONNECTION Configuration Parameter

onconfig.std *value*

None

range of values

0, 1, 2 0 = no security checking occurs 1 = Informix checks whether the ID of the user who is running the program matches the ID of the user who is trying to connect to the database. 2 = same as 1, plus Informix retrieves the peer port number from the network API and verifies that the connection is coming from the client program. You can only specify 2 if your system has SOCTCP or IPCSTR network protocols.

takes effect

When the database server is shut down and restarted

refer to Role of the SERVERNUM configuration parameter, in the multiple-residency chapter of the *IBM Informix Administrator's Guide*

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

SEQ_CACHE_SIZE configuration parameter

Use the SEQ_CACHE_SIZE configuration parameter to specify the maximum number of sequence objects that can have preallocated values in the sequence cache.

If SEQ_CACHE_SIZE configuration parameter is not set, then by default you cannot define more than 10 different sequence objects with the CACHE option of the CREATE SEQUENCE statement.

onconfig.std *value*
10

default value
10

range of values
1 to 2147483647

takes effect
When the database server is stopped and restarted

For example, if you want to increase the maximum number of sequence objects that can have preallocated values to 15, specify:

SEQ_CACHE_SIZE 15

SERVERNUM configuration parameter

The SERVERNUM configuration parameter specifies a relative location in shared memory.

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

onconfig.std *value*
0

range of values
0 through 255

takes effect
When the database server is shut down and restarted

refer to Role of the SERVERNUM configuration parameter, in the multiple-residency chapter of the *IBM Informix Administrator's Guide*

SHMADD configuration parameter

Use the SHMADD configuration parameter to specify the size of a segment that is dynamically added to the virtual portion of shared memory.

onconfig.std *value*
8192

range of values
32-bit platforms: 1024 through 524288 64-bit platforms: 1024 through 4294967296

units Kilobytes

takes effect
When the database server is shut down and restarted

utilities
onstat -g seg (Use the **onstat -g seg** command to display the number of

shared-memory segments that the database server is currently using. For more information, see “**onstat -g seg** command: Print shared memory segment statistics” on page 20-161)

refer to The following material in the *IBM Informix Administrator's Guide*:

- Virtual portion of shared memory, in the shared-memory chapter
- Monitoring shared-memory segments with **onstat -g seg**, in the managing memory chapter

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

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

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

You can dynamically change the value of the SHMADD configuration parameter by using the **onmode -wm** or **onmode -wf** command.

SHMBASE Configuration Parameter

onconfig.std *value*

Platform dependent

units Address

range of values

Positive integers

takes effect

When the database server is shut down and restarted

utilities

To see the shared-memory segment addresses, use the **onstat -g seg** command.

refer to Setting operating-system shared-memory configuration parameters, in the chapter on managing shared memory in the *IBM Informix Administrator's Guide*

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

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

SHMNOACCESS Configuration Parameter

onconfig.std *value*

On UNIX: None

On Windows: #SHMNOACCESS 0x70000000-0x7FFFFFFF, and this value is commented out in the onconfig.std template file.

separators

Comma

range of values

From one to ten address ranges

takes effect

When the database server is shut down and restarted

The SHMNOACCESS configuration parameter specifies a virtual memory address range to **not** use to attach shared memory. SHMNOACCESS is used to avoid specific range process addresses, which in turn avoids conflicts with operating system libraries.

Each address in each range must start in hexadecimal format. Each address in a range must be separated by a hyphen and each range must be separated by a comma, as the following example shows:

```
SHMNOACCESS 0x70000000-0x75000000,  
0x7A000000-0x80000000
```

SHMTOTAL configuration parameter

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

onconfig.std *value*

0

units Kilobytes

range of values

0 (= no specific limit) or any integer greater than or equal to 1

takes effect

When the database server is shut down and restarted

refer to How much shared memory the database server needs, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

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

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

This message includes the following values.

Value	Description
-------	-------------

xx	Current size of resident segments
----	-----------------------------------


yy Current size of virtual segments
zz Total shared memory required

If you enabled the LOW_MEMORY_MGR configuration parameter and are configuring the server to use a percentage of the SHMTOTAL value for automatic low memory management start and stop thresholds, use caution when changing the value of SHMTOTAL. Changing the value of the SHMTOTAL configuration parameter value can cause the configuration of automatic low memory management to become invalid, forcing the database server to use the default settings.

UNIX Only:

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

Related concepts:

 Shared-memory size (Administrator's Guide)

Related reference:

“LOW_MEMORY_MGR configuration parameter” on page 1-89

“scheduler Imm enable argument: Specify automatic low memory management settings (SQL administration API)” on page 21-101

SHMVIRT_ALLOCSEG configuration parameter

onconfig.std *value*
0,3

units First parameter: a decimal number indicating the percentage of memory used before a segment is added OR the whole number of kilobytes remaining in the server when a segment is added.

Second parameter: alarm level

range of values

First parameter: 0 (the default) OR any percentage from .40 to .99 OR a whole number of kilobytes from 256 to 1,000,000 = Disabled

Second parameter: Alarm level values from 1 to 5 where: 1 = Not noteworthy, 2 = Information, 3 = Attention, 4 = Emergency, 5 = Fatal

takes effect

When the database server is shut down and restarted

refer to

- “Event Alarm Parameters” on page C-4
- Configuration Parameters That Affect Memory Utilization in the *IBM Informix Performance Guide*

SHMVIRT_ALLOCSEG specifies a threshold at which Informix should allocate a new memory segment and the alarm level activated if the server cannot allocate the new memory segment, thus ensuring that the server never runs out of memory. Once the alarm level is activated, it will repeat every thirty minutes if a new memory segment cannot be allocated.

SHMVIRTSIZE configuration parameter

Use the SHMVIRTSIZE configuration parameter to specify the initial size of a virtual shared-memory segment.

onconfig.std *value*

32656

if not present

- If SHMADD is present: SHMADD
- If SHMADD is not present: 8192

units Kilobytes

range of values

32-bit platforms: Positive integer with a maximum value of 2 gigabytes

64-bit platforms: Positive integer with a maximum value of 4 terabytes

The maximum value might be less on some platforms due to operating-system limitations. For the actual maximum value for your UNIX platform, see the machine notes.

takes effect

When the database server is shut down and restarted

utilities

onstat -g seg (see “**onstat -g seg** command: Print shared memory segment statistics” on page 20-161)

refer to

- Virtual portion of shared memory, in the shared-memory chapter of the *IBM Informix Administrator's Guide*
- Chapter on configuration effects on memory utilization, in your *IBM Informix Performance Guide*

To determine the appropriate value for the SHMVIRTSIZE configuration parameter, use the following algorithm to determine the size of the virtual portion of shared memory:

$$\text{shmvirtsize} = \text{fixed overhead} + ((\text{stack size} + \text{heap}) * \text{number of users})$$

Component	Value to use
Fixed overhead	<p>This includes the size of the AIO vectors + sort memory + dbspace backup buffers + dictionary size + size of stored-procedure cache + histogram pool + other pools, and other overhead.</p> <p>To obtain an estimate of the fixed overhead, start the database server and see how many additional memory segments are allocated, if any. The number of users that you have on the system when you start the server, impacts the allocation of memory segments. When you start the server:</p> <ul style="list-style-type: none"> • If the number of users is typical for your environment, then add the size of the memory segments to the current value for the SHMVIRTSIZE configuration parameter and restart the server. • If the number of users is far less than what is typical for your environment, you must calculate the appropriate overhead value to use for the memory segments. You can determine how many memory segments each user will consume by dividing the number of additional memory segments allocated when you started the server by the number of users that you had on the server at that time. Multiply the value for the memory segments for each user by the number of users that you will typically have on the system. Add this calculated value for the memory segments to the current value for SHMVIRTSIZE configuration parameter and restart the server.
Stack size	<p>On 32-bit systems, use 32 kilobytes for the stack size. Typically on 64-bit systems, you use 64 kilobytes for the stack size. However, some 64-bit systems use a different value. For more information see the STACKSIZE configuration parameter.</p>
Heap	Use 30 kilobytes per user.
Number of users	Use the maximum number of concurrent user sessions that you anticipate on the server.

If possible, create a virtual portion of shared memory of a size that is more than you require for your daily processing.

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

Related reference:

“LOW_MEMORY_RESERVE configuration parameter” on page 1-90

SINGLE_CPU_VP configuration parameter

onconfig.std value

0

range of values

0 = running with multiple CPU VPs

Any nonzero value = running with one CPU VP

takes effect

When the database server is shut down and restarted

refer to Running on a single-processor computer, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*

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

Disable the SINGLE_CPU_VP configuration parameter by setting it to 0 if you want the number of CPU VPs to be automatically increased when the database server starts.

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

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

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


onmode: failed when trying to change the number of *classname* VPs by *n*.

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

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

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

Related concepts:

 Automatic addition of CPU virtual processors (Performance Guide)

VPCLASS Values and the SINGLE_CPU_VP Configuration Parameter

Informix treats user-defined virtual-processor classes as if they were CPU virtual processors. If you set the *SINGLE_CPU_VP* configuration parameter to a nonzero value, you cannot create any user-defined virtual-processor classes.

Using a user-defined VPCLASS

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

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

Using the *cpu* VPCLASS

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

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

SMX_COMPRESS Configuration Parameter

Use the SMX_COMPRESS configuration parameter to specify the level of compression that the database server uses before sending data from the source database server to the target database server.

Network compression saves network bandwidth over slow links but uses more CPU to compress and decompress the data. The SMX_COMPRESS configuration parameter values of the two servers are compared and changed to the higher compression values.

onconfig.std *value*

0

range of values

- -1 specifies no compression
- 0 specifies that data is compressed only if the target server expects compression
- 1 through 9 specifies increasing levels of compression

takes effect

When the database server is shut down and restarted

refer to The **onmode -wf** command

The values have the following meanings:

Value	Meaning
-1	The source database server never compresses the data, regardless of whether or not the target site uses compression.
0	The source database server compresses the data only if the target database server expects compressed data.
1	The database server performs a minimum amount of compression.
9	The database server performs the maximum possible compression.

SP_AUTOEXPAND configuration parameter

Use the SP_AUTOEXPAND configuration parameter to enable or disable the automatic creation or extension of chunks.

onconfig.std *value*

1

default value

1

range of values

0 = The automatic creation or extension of chunks is not enabled.

1 = The automatic creation or extension of chunks is enabled.

takes effect

When the database server is stopped and restarted

When the SP_AUTOEXPAND configuration parameter is enabled and a storage container such as a dbspace has a defined create size or extend size that is not zero, the container is auto-expandable.

You can dynamically change the value of the SP_AUTOEXPAND configuration parameter by using the **onmode -wm** or **onmode -wf** command.

SP_THRESHOLD Configuration Parameter

Use the SP_THRESHOLD configuration parameter to define the minimum amount of free kilobytes that can exist in a storage space before Informix automatically runs a task to expand the space, either by extending an existing chunk in the space or by adding a new chunk.

onconfig.std *value*

0

default value

0

range of values

0 = No threshold. The trigger that runs the storage space monitoring (**mon_low_storage**) task for adding space when space is below the threshold is disabled.

1 to 50 = A threshold that is a percentage of free kilobytes in a storage space.

If the value is 50 or below, Informix interprets the value as a percentage (for example, 10 = 10 percent and 2.84 = 2.84 percent).

1000 to the maximum size of a chunk = A threshold that is either 1000 kilobytes or the maximum size of the chunk on the current platform.

If the value is 1000 or higher, Informix interprets the value as a specific number of kilobytes.

takes effect

When the database server is stopped and restarted

Values between 50 and 1000 are not valid.

When you set the SP_THRESHOLD configuration parameter to a valid value that is greater than 0, the built-in Scheduler task, **mon_low_storage**, runs automatically when the free space in a dbspace, temporary dbspace, sbspace, temporary sbspace, or blobspace falls below the threshold.

Suppose the value of the SP_THRESHOLD configuration parameter value is 5.5, which the server interprets as 5.5 percent. If a space runs low on free pages, and the free space percentage falls below 5.5 percent and remains below that level until the **mon_low_storage** task runs next, that task will attempt to expand the space. If the SP_THRESHOLD configuration parameter is set to 50000 and a space has fewer than 50000 free kilobytes, that space will be expanded the next time **mon_low_storage** task runs.

A value of 0 turns off the **mon_low_storage** task, and prevents the server from extending any space. However, a value of 0 does not affect the ability of the server to extend a space when all free pages are depleted and more are needed.

The value specified in the SP_THRESHOLD configuration parameter applies to all spaces belonging to the server.

You can dynamically change the value of the SP_THRESHOLD configuration parameter by using the **onmode -wm** or **onmode -wf** command.

SP_WAITTIME Configuration Parameter

Use the SP_WAITTIME configuration parameter to specify the maximum number of seconds that a thread waits for a dbspace, temporary dbspace, sbspace, temporary sbspace, or blobspace space to expand before returning an out-of-space error.

onconfig.std *value*

30

units seconds

default value

30

range of values

0 to 2147483647

takes effect

When the database server is stopped and restarted

The time that the server uses to automatically add or expand a chunk can vary widely, depending on various factors such as the size of the chunk, the speed of the associated disk drives, and the load on the system. When Informix automatically adds or expands a chunk to prevent free space from falling below the threshold specified by the SP_THRESHOLD configuration parameter, Informix forces threads that need the space to wait until it is available. You can change the value of the SP_WAITTIME configuration parameter if you want to change the maximum amount of time that the thread will wait for more space.

A thread will wait for a storage space to expand only if the storage pool contains entries. A thread will not wait if the storage pool is empty.

You can dynamically enable or disable the wait-time value by using the **onmode -wm** or **onmode -wf** command.

SQL_LOGICAL_CHAR Configuration Parameter

Use the SQL_LOGICAL_CHAR configuration parameter to enable or disable the expansion of size specifications in declarations of built-in character data types.

onconfig.std *value*

OFF (= interpret size specifications in units of bytes)

range of values

OFF, 1, 2, 3, 4, ON,

takes effect

When the database server is shut down and restarted

refer to

- Information about the **flags** column of the **systables** system catalog table, in the *IBM Informix Guide to SQL: Reference*
- Enabling logical character semantics in character data type declarations, in the *IBM Informix Guide to SQL: Syntax*
- Multibyte code sets and logical character support in SQL features, in the *IBM Informix GLS User's Guide*

For applications that are developed in single-byte locales, but deployed in multibyte locales, this feature can reduce the risk of multibyte logical characters being truncated during data entry operations.

In a multibyte code set, such as **UTF-8** or the multibyte code sets for some East Asian languages, a single logical character can require more than one byte of storage. The setting of this parameter can instruct the SQL parser to apply logical-character semantics to declarations of these built-in character data types:

- CHAR
- CHARACTER
- CHARACTER VARYING
- LVARCHAR
- NCHAR
- NVARCHAR
- VARCHAR
- DISTINCT types that declare any of these data types as their base types
- ROW types (named and unnamed) that include fields of these data types
- Collection types (LIST, MULTISSET, or SET) that include these types as elements.

The setting that you specify for this parameter must be one of the following values:

Value	Effect
OFF	No expansion of declared sizes
1	No expansion of declared sizes
2	Use 2 as the expansion factor for declared sizes.
3	Use 3 as the expansion factor for declared sizes.
4	Use 4 as the expansion factor for declared sizes.

Value	Effect
ON	Use <i>M</i> as the expansion factor, where <i>M</i> is the maximum length in bytes that any logical character requires in the code set of the current database. Depending on the DB_LOCALE setting, <i>M</i> has an integer range from 1 (in single-byte locales) up to 4.

Whether the SQL_LOGICAL_CHAR configuration parameter is set to enable or disable the expansion of declared storage sizes, its setting specifies how data type declarations are interpreted for all sessions of the Informix instance.

Automatic Resizing of the Expansion Factor

When SQL_LOGICAL_CHAR is set to a valid digit, and the current session creates a database, Informix compares the SQL_LOGICAL_CHAR value with the maximum number of bytes that any logical character will use for the code set of the database.

If the SQL_LOGICAL_CHAR setting is greater than that maximum number of bytes, the database uses the maximum value for the locale as the new expansion factor, overriding what the configuration file specifies. The SQL_LOGICAL_CHAR setting in the configuration file remains unchanged, and continues to act as the default expansion factor for other user databases.

Similarly, if the SQL_LOGICAL_CHAR value for a session is automatically reset to a digit, as described above, but the same session subsequently connects to another database whose locale uses a code set in which a logical character requires a larger storage size than the current expansion factor, Informix uses the maximum number of bytes for the new code set as the new expansion factor while the user session is connected to that database, rather than using the current setting of SQL_LOGICAL_CHAR.

Automatic resetting of the expansion factor to match the largest logical character size in the code set that **DB_LOCALE** specifies at connection time also occurs when SQL_LOGICAL_CHAR is set to ON, but the effects of the ON setting are not identical to the database server behavior when SQL_LOGICAL_CHAR is set to a digit (1, 2, 3, or 4) in two ways:

- The expansion factor can be automatically reset to a smaller value if ON is the SQL_LOGICAL_CHAR setting.
- There is no difference between SQL_LOGICAL_CHAR = 4 and SQL_LOGICAL_CHAR = ON.

You must set SQL_LOGICAL_CHAR to ON, rather than to a digit, if you want a smaller expansion factor when the current session connects to a database whose largest logical character in the **DB_LOCALE** code set requires a smaller number of bytes than the current SQL_LOGICAL_CHAR setting. The effective expansion factor will always be less than or equal to the maximum character size for a locale.

SQLTRACE Configuration Parameter

syntax

```
SQLTRACE [level=low|medium|high|off],
[ntraces=number of traces],
[size=size of each trace buffer]
,[mode=global|user]
```

onconfig.std *value*

#SQLTRACE level=low,ntraces=1000,size=2,mode=global (suggested value, but not set)

range of values

level Amount of information traced:

- Low. (Default) Captures statement statistics, statement text, and statement iterators.
- Medium. Captures all of the information included in low-level tracing, plus table names, the database name, and stored procedure stacks.
- High. Captures all of the information included in medium-level tracing, plus host variables.
- Off. Specifies no SQL tracing.

ntraces

Number of SQL statements to trace before reusing the resources. The range is 500 to 2147483647.

size Maximum size of variable length data to be stored. The range is 1 Kilobyte to 100 Kilobytes. If this buffer size is exceeded, the database server discards saved data.

mode Type of tracing performed:

- Global. For all users on the system
- User. For users who have tracing enabled by an SQL administration API **task()** or **admin()** function. (Specify this mode if you want to get a sample of the SQL that a small set of users is running.)

takes effect

When the database server is shut down and restarted

utilities

onstat -g his

refer to SQL tracing information in the *IBM Informix Administrator's Guide*

The SQLTRACE parameter controls the startup environment of SQL tracing.

SSL_KEYSTORE_LABEL Configuration Parameter

onconfig.std *value*

None

range of values

Up to 512 characters for the label of the Informix certificate used in Secure Sockets Layer (SSL) protocol communications

takes effect

When the database server is shut down and restarted

refer to Secure Sockets Layer protocol information in the *IBM Informix Security Guide*

The SSL_KEYSTORE_LABEL configuration parameter specifies the label of the server digital certificate used in the keystore database, a protected database that stores SSL keys and digital certificates. The default value is name of the label for

the default SSL certificate that is stored in the Informix keystore in the `INFORMIXDIR/ssl/servername.kdb` directory.

For information on configuration parameters that you need to set on clients, see the IBM Informix Security Guide.

STACKSIZE configuration parameter

Use the STACKSIZE configuration parameter to specify the stack size for the database server user threads.

onconfig.std *value*

32 for 32-bit database servers 64 for 64-bit database servers

units Kilobytes

range of values

32 through limit determined by the database server configuration and the amount of memory available

takes effect

When the database server is shut down and restarted

refer to

- Stacks, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*
- CREATE FUNCTION statement, in the *IBM Informix Guide to SQL: Syntax*

The value of STACKSIZE does not have an upper limit, but setting a value that is too large wastes virtual memory space and can cause swap-space problems.

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

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

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

You can dynamically change the value of the STACKSIZE configuration parameter by using the **onmode -wm** or **onmode -wf** command.

Related reference:

 INFORMIXSTACKSIZE environment variable (SQL Reference)

STAGEBLOB Configuration Parameter

onconfig.std *value*

None

range of values

Up to 128 bytes. STAGEBLOB must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or \$ characters.

takes effect

When the database server is shut down and restarted

refer to IBM Informix Optical Subsystem Guide

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

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

STATCHANGE configuration parameter

Use the STATCHANGE configuration parameter to specify a positive integer for a global percentage of a change threshold for the server to use to determine if distribution statistics qualify for an update when the automatic mode for UPDATE STATISTICS operations is enabled.

onconfig.std *value*

10

units percentage of a change threshold

default value

10

range of values

0 through 100

takes effect

When the database server is stopped and restarted

Informix uses the value of the STATCHANGE configuration parameter when the AUTO_STAT_MODE configuration parameter, or session environment setting, or the AUTO keyword of the UPDATE STATISTICS statement has enabled the automatic mode for UPDATE STATISTICS operations.

The STATCHANGE setting specifies a change threshold for the database server to use to determine if distribution statistics qualify for an update when the automatic mode for UPDATE STATISTICS operations is enabled. When this mode is enabled, the UPDATE STATISTICS statement compares the STATCHANGE setting with the percentage of rows that have changed in each table or fragment since the current data distributions were calculated, and selectively updates only the missing or stale distribution statistics for each table or fragment within the scope of the UPDATE STATISTICS statement.

Related concepts:

 STATCHANGE Environment Option (SQL Syntax)

Related reference:

“AUTO_STAT_MODE configuration parameter” on page 1-33

 Statistics options of the CREATE TABLE statement (SQL Syntax)

STMT_CACHE Configuration Parameter

onconfig.std *value*

0

if not present

0

range of values

0, 1, or 2

takes effect

When the database server is shut down and restarted

utilities

onmode -e

refer to

- “onmode -e: Change usage of the SQL statement cache” on page 14-13
- Using the SQL statement cache, in the *IBM Informix Performance Guide*

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

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

The following table describes the possible values.

Possible Value

Meaning

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

Related reference:

 STMT_CACHE environment variable (SQL Reference)

STMT_CACHE_HITS Configuration Parameter

onconfig.std *value*

0

if not present

0

units Integer

range of values

Any value greater than or equal to 0

takes effect

When the database server is shut down and restarted

utilities

onmode -W STMT_CACHE_HITS onstat (See “onstat -g ssc command: Print SQL statement occurrences” on page 20-174.)

refer to

- “onmode -W: Change settings for the SQL statement cache” on page 14-24
- Improving query performance, in the *IBM Informix Performance Guide*

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

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

STMT_CACHE_NOLIMIT Configuration Parameter

onconfig.std value
0

if not present
1

range of values
0 or 1

takes effect
When the database server is shut down and restarted

utilities
onmode -W STMT_CACHE_NOLIMITonstat (See “onstat -g ssc command: Print SQL statement occurrences” on page 20-174.)

refer to

- “onmode -W: Change settings for the SQL statement cache” on page 14-24
- Improving query performance, in the *IBM Informix Performance Guide*

STMT_CACHE_NOLIMIT controls whether to insert qualified statements into the SQL statement cache. The following table describes the possible values.

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

STMT_CACHE_NUMPOOL Configuration Parameter

onconfig.std *value*

1

if not present

1

units Positive integer

range of values

1 to 256

takes effect

When the database server is shut down and restarted

refer to Improving query performance, in the *IBM Informix Performance Guide*

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

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

STMT_CACHE_SIZE Configuration Parameter

onconfig.std *value*

512

default size of SQL statement cache

512 kilobytes (524288 bytes)

units Kilobytes

range of values

Positive integer

takes effect

When the database server is shut down and restarted

utilities

onstat -g ssc (Maxsize field)

refer to Improving query performance, in the *IBM Informix Performance Guide*

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

STOP_APPLY Configuration Parameter

Use the STOP_APPLY configuration parameter to stop an RS secondary server from applying log files received from the primary server.

Stopping the application of log files allows you to recover quickly from erroneous database modifications by restoring the data from the RS secondary server. You can configure the server to either stop the application of logs immediately, or at a specified point in time. When setting the value of STOP_APPLY you must also set

LOG_STAGING_DIR. If STOP_APPLY is configured and LOG_STAGING_DIR is not set to a valid and secure directory, the server cannot be initialized.

onconfig.std *value*

None

if not present

0 (Apply logs)

range of values (first parameter)

0, 1, "YYYY:MM:DD-hh:mm:ss". 0 = Apply logs, 1 = Stop applying logs immediately, "YYYY:MM:DD-hh:mm:ss" = Stop the log apply at a specified time, where:

- YYYY = Year
- MM = Month
- DD = Day
- hh = hour
- mm = minute
- ss = second

takes effect

when the database server is shut down and restarted or when the parameter is reset dynamically using **onmode -wf**.

utilities

onmode -wf or **onmode -wm**

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- “DELAY_APPLY Configuration Parameter” on page 1-53
- “LOG_STAGING_DIR Configuration Parameter” on page 1-87
- “UPDATABLE_SECONDARY Configuration Parameter” on page 1-152
- *RS Secondary Server Latency for Disaster Recovery* in the *IBM Informix Administrator's Guide*

Log files are stored in binary format in a directory specified by the LOG_STAGING_DIR configuration parameter. You must specify a valid and secure location for the log files. See “LOG_STAGING_DIR Configuration Parameter” on page 1-87. Also see the “DELAY_APPLY Configuration Parameter” on page 1-53

The time value specified for the STOP_APPLY configuration parameter is assumed to be in the same timezone as the RS secondary server.

The **dbexport** utility cannot support write operations on an updatable secondary server unless the STOP_APPLY parameter is set. (Besides STOP_APPLY, the UPDATABLE_SECONDARY and USELASTCOMMITTED configuration parameters must also be set to enable write operations by **dbexport** on a secondary data replication server.)

Related reference:

“**onstat -g cluster** command: Print high-availability cluster information” on page 20-65

STORAGE_FULL_ALARM Configuration Parameter

onconfig.std *value*
600,3

units *seconds,severity_level*

range of values

seconds = 0 (off) or a positive integer indicating the number of seconds between notifications

severity_level = 0 (no alarms) or 1 through 5

takes effect

When the database server is shut down and restarted

utilities

None

refer to The section on how to monitor storage spaces is in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

Use the STORAGE_FULL_ALARM configuration parameter to configure the frequency and severity of messages and alarms when storage spaces become full. When a storage space, such as a dbspace, sbspace, blobspace, or tblspace, or a partition becomes full, an alarm is raised and a message is sent to the online message log. You can specify the number of seconds between notifications with the first value of this parameter. You can specify the lowest severity for event alarms to be returned. Setting a specific severity prevents events that have a lower severity from being raised. But events that have the same or greater severity as the severity specified are raised. For more information on alarm levels, see “Event Alarm Parameters” on page C-4. You can prevent alarms when storage spaces become full by setting this parameter to 0.

Regardless of the value of STORAGE_FULL_ALARM, messages are sent to the online message log when storage spaces or partitions become full.

SYSALARMPROGRAM configuration parameter

Use the SYSALARMPROGRAM configuration parameter to specify the full pathname of the **evidence.sh** script.

The database server executes **evidence.sh** when a database server failure occurs. The output from the **evidence.sh** script can be used to diagnose the cause of a database server failure.

onconfig.std *value*

On UNIX: **\$INFORMIXDIR/etc/evidence.sh**

On Windows: Not set. (Commented out.) Listed as **\$INFORMIXDIR\etc\evidence.bat**

range of values

Pathname

takes effect

When the database server is shut down and restarted

utilities

None

refer to None

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

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

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

Table 1-57. Enabling command extensions from the Windows registry

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

You can dynamically change the value of the SYSALARMPROGRAM configuration parameter by using the **onmode -wm** or **onmode -wf** command.

SYSSBSPACENAME Configuration Parameter

onconfig.std *value*

None

if not present

0

range of values

Up to 128 bytes. SYSSBSPACENAME must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or \$ characters.

takes effect

When disk is initialized (destroys all data)

utilities

onspaces

refer to

- “onspaces -c -S: Create an sbspace” on page 19-11
- “Sbspace Structure” on page 4-24
- Updating statistics, in the chapter on individual query performance in your *IBM Informix Performance Guide*
- Sbspace characteristics, in the chapter on configuration effects on I/O in your *IBM Informix Performance Guide*
- Writing user-defined statistics, in the performance chapter in *IBM Informix User-Defined Routines and Data Types Developer's Guide*

- Providing statistics data for a column, in the *IBM Informix DataBlade API Programmer's Guide*
- "SBSPACENAME Configuration Parameter" on page 1-121 (specifies the name of the default sbspace)

SYSSBSPACENAME specifies the name of the sbspace in which the database server stores fragment-level data-distribution statistics, which the **syfragsdist** system catalog table stores as BLOB objects in its **encsdist** column. To support fragment level statistics, you must specify the name of an sbspace as the SYSSBSPACENAME setting, and you must allocate that sbspace (by using the **onspaces** utility, as described below. For any table whose STATLEVEL attribute is set to FRAGMENT, the database server returns an error if SYSSBSPACENAME is not set, or if the sbspace to which SYSSBSPACENAME is set was not properly allocated).

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

Do not confuse the SYSSBSPACENAME configuration parameter with the SBSPACENAME configuration parameter .

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

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

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

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

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

TBLSPACE_STATS configuration parameter

Use the TBLSPACE_STATS configuration parameter to turn on and off the collection of tblspace statistics.

onconfig.std *value*
1

if not present

1

units Integer

range of values

0 or 1

takes effect

When the database server is shut down and restarted

Use the **onstat -g ppf** command to list tblspace statistics.

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

To turn on the collection of tblspace statistics, set TBLSPACE_STATS to 1.

TBLTBLFIRST configuration parameter

Use the TBLTBLFIRST configuration parameter to specify the first extent size of tblspace **tblspace** in the root dbspace.

onconfig.std *value*

0

units Kilobytes in multiples of page size

range of values

From the equivalent of 250 pages specified in kilobytes to the size of the first chunk minus the space needed for any system objects.

takes effect

When the database server is initialized

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

You can use **oncheck -pt** and **oncheck -pT** to show the first and next extent sizes of a tblspace **tblspace**. For more information about the **oncheck** utility, see "oncheck -pt and -pT: Display tblspaces for a Table or Fragment" on page 8-19.

If you want to configure the first extent for a non-root dbspace, see information about the **onspaces** utility in Chapter 19, "The onspaces Utility," on page 19-1.

You can dynamically change the value of the TBLTBLFIRST configuration parameter by using the **onmode -wm** or **onmode -wf** command.

TBLTBLNEXT configuration parameter

The TBLTBLNEXT configuration parameter specifies the next extent size of tblspace **tblspace** in the root dbspace.

onconfig.std *value*

0

units Kilobytes

range of values

From equivalent of 4 pages specified in kilobytes to the maximum chunk size minus three pages

takes effect

When the database server is initialized

If there is not enough space for a next extent in the primary chunk, the extent is allocated from another chunk. If the specified space is not available, the closest available space is allocated. For more information on configuring extent sizes in `tblspace` **tblspace**, see “TBLTBLFIRST configuration parameter” on page 1-150.

You can dynamically change the value of the TBLTBLNEXT configuration parameter by using the **onmode -wm** or **onmode -wf** command.

TEMPTAB_NOLOG configuration parameter

Use the TEMPTAB_NOLOG configuration parameter to disable logging on temporary tables.

This parameter can improve performance in application programs because it prevents Informix from transferring temporary tables over the network. The setting can be updated dynamically with the **onmode -wf** utility.

If you enable this setting, be aware that because no data is logged when using temporary tables, rolling back a transaction on a temporary table will no longer undo the work in the temporary table.

For HDR, RSS, and SDS secondary servers in a high-availability cluster, logical logging on temporary tables should always be disabled by setting the TEMPTAB_NOLOG configuration parameter to 1.

onconfig.std *value*
0

takes effect

When the database server is shut down and restarted

range of values

0 = Enable logical logging on temporary table operations
1 = Disable logical logging on temporary table operations

utilities

onmode -wf

TXTIMEOUT configuration parameter

Use the TXTIMEOUT configuration parameter to specify the amount of time that a participant in a two-phase commit waits before it initiates participant recovery.

onconfig.std *value*
300

units Seconds

range of values

Positive integers

takes effect

When the database server is shut down and restarted

refer to How the two-phase commit protocol handles failures, in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*

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

You can dynamically change the value of the TXTIMEOUT configuration parameter by using the **onmode -wm** or **onmode -wf** command.

UNSECURE_ONSTAT Configuration Parameter

onconfig.std *value*

None

possible values

1 = All users can run **onstat** commands to view running SQL statements

takes effect

When the database server is shut down and restarted

refer to *IBM Informix Administrator's Guide*

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

UPDATABLE_SECONDARY Configuration Parameter

Use the UPDATABLE_SECONDARY configuration parameter to set the number of connections to establish between the primary and secondary servers. Setting this configuration parameter enables client applications to perform update, insert, and delete operations on a high-availability secondary server.

onconfig.std *value*

0

if not present

0

units Number of network connections between a given secondary server and its primary server

range of values

Any number from zero (the default value) up to the total number of network connections

takes effect

When the secondary database server is shut down and restarted

refer to

- Data Replication Overview chapter of the *IBM Informix Administrator's Guide*
- Server Multiplexer Group (SMX) Connections chapter of the *IBM Informix Administrator's Guide*

Isolation Levels for Secondary Data Replication Servers

If the `UPDATABLE_SECONDARY` configuration parameter is not set or is set to zero (0), a secondary data replication server is read-only. In this case, only the `DIRTY READ` or `READ UNCOMMITTED` transaction isolation levels are available on secondary servers.

If the `UPDATABLE_SECONDARY` parameter is set to a valid number of connections greater than zero (0), a secondary data replication server can support the `COMMITTED READ`, `COMMITTED READ LAST COMMITTED`, or `COMMITTED READ` transaction isolation level, or the `USELASTCOMMITTED` session environment variable. Only SQL DML statements, such as `INSERT`, `UPDATE`, `MERGE`, and `DELETE`, and the **dbexport** utility, can support write operations on an updatable secondary server. (Besides `UPDATABLE_SECONDARY`, the `STOP_APPLY` and `USELASTCOMMITTED` configuration parameters must also be set to enable write operations by **dbexport** on a secondary data replication server.)

Related reference:

“onstat -g cluster command: Print high-availability cluster information” on page 20-65

USELASTCOMMITTED Configuration Parameter

Use the `USELASTCOMMITTED` configuration parameter to specify the isolation level for which the `LAST COMMITTED` feature of the `COMMITTED READ` isolation level is implicitly in effect.

The `LAST COMMITTED` feature can reduce the risk of locking conflicts between concurrent transactions on tables that have exclusive row locks. The `USELASTCOMMITTED` configuration parameter can also enable `LAST COMMITTED` semantics for `READ COMMITTED` and `READ UNCOMMITTED` isolation levels of the `SET TRANSACTION` statement.

The `USELASTCOMMITTED` configuration parameter only works with tables that have been created or altered to have `ROW` as their locking granularity. Tables created without any explicit lock mode setting will use the default setting in `DEF_TABLE_LOCKMODE`. If `DEF_TABLE_LOCKMODE` is set to `PAGE`, the `USELASTCOMMITTED` configuration parameter cannot enable access to the most recently committed data in tables on which uncommitted transactions hold exclusive locks, unless the tables were explicitly altered to have `ROW` level of locking granularity. For more details, please refer to “`DEF_TABLE_LOCKMODE` configuration parameter” on page 1-52.

onconfig.std *value*

None

range of values

None = No isolation level identified. If your session encounters an exclusive lock when attempting to read a row in the Committed Read, Dirty Read, Read Committed, or Read Uncommitted isolation level, your transaction cannot read that row until the concurrent transaction that holds the exclusive lock is committed or rolled back.

'Committed Read'

= All transactions from a Committed Read isolation level are

treated as last committed transactions. The database server reads the most recently committed version of the data when it encounters an exclusive lock while attempting to read a row in the Committed Read or Read Committed isolation level.

'Dirty Read'

= All transactions from a Dirty Read isolation level are treated as last committed transactions. The database server reads the most recently committed version of the data if it encounters an exclusive lock while attempting to read a row in the Dirty Read or Read Uncommitted isolation level.

All = All transactions from both Committed Read and Dirty Read isolation levels are treated as last committed transactions. database server reads the most recently committed version of the data if it encounters an exclusive lock while attempting to read a row in the Committed Read, Dirty Read, Read Committed, or Read Uncommitted isolation level.

takes effect

When the database server is stopped and restarted.

utilities

onmode -wf or -wm

refer to

- “onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25
- SET ISOLATION statement information in *IBM Informix Guide to SQL: Syntax*
- Isolation level information in *IBM Informix Performance Guide*

Use with Shared Disk Secondary Database Servers

The USELASTCOMMITTED configuration parameter is also valid on SD (Shared Disk) secondary database servers. The following table shows valid values for the USELASTCOMMITTED configuration parameter on SD secondary servers and their descriptions.

Table 1-58. Valid Secondary Server USELASTCOMMITTED Values

USELASTCOMMITTED value	Description
None	COMMITTED READ LAST COMMITTED is not the default isolation level for sessions
Committed Read	COMMITTED READ LAST COMMITTED is the default isolation level for all sessions with Committed Read isolation
Dirty Read	COMMITTED READ LAST COMMITTED is the default isolation level for all sessions with Dirty Read isolation
All	COMMITTED READ LAST COMMITTED is the default isolation level for all sessions with Committed Read or Dirty Read isolation

You can dynamically change the values of the USELASTCOMMITTED configuration parameter through the SET ISOLATION statement or by using **onmode -wf** or **onmode -wm**.

Related concepts:

 USELASTCOMMITTED Environment Option (SQL Syntax)

USEOSTIME Configuration Parameter

onconfig.std *value*

0

range of values

0 = Off 1 = On

takes effect

During initialization

refer to

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

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

2001-09-29 12:50:04.612

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

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

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

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

USTLOW_SAMPLE configuration parameter

Use the USTLOW_SAMPLE configuration parameter to enable the generation of index statistics based on sampling when you run UPDATE STATISTICS statements in LOW mode.

onconfig.std *value*

0

default value

0

range of values

0 = disable sampling

1 = enable sampling

utility commands

onmode -wm or **onmode -wf**

takes effect

When the database server is stopped and restarted or when the USTLOW_SAMPLE value is changed by an **onmode -wm** or **onmode -wf** command.

For an index with more than 100 K leaf pages, the gathering of statistics using sampling can increase the speed of the UPDATE STATISTICS operation.

Related concepts:

➞ USTLOW_SAMPLE environment option (SQL Syntax)

➞ Data sampling during update statistics operations (Performance Guide)

VP_MEMORY_CACHE_KB Configuration Parameter

onconfig.std *value*

0

range of values

0 = Off 800 kilobytes per VP to the total kilobytes of the private caches in all VPs, not to exceed 40% of the memory limit as specified in the SHMTOTAL configuration parameter

takes effect

During initialization

utilities

onmode -wf or **-wm onstat -g vpcache**

refer to the CPU Virtual Processor Memory Caches section of the *IBM Informix Performance Guide*

The VP_MEMORY_CACHE_KB parameter enables the database server to access the private memory blocks of your CPU VP.

Informix uses the VP private memory cache when a thread needs to allocate whole memory blocks. Informix does not use the private memory cache for buffers in the buffer pool or for any pages read from disk. For example, if you set the VP_MEMORY_CACHE_KB configuration parameter to 1000, the sum of all memory blocks in the cache will be no greater than 1000 kilobytes in size.

VP_MEMORY_CACHE_KB can be changed using the **onmode -wf** or **-wm** options; the changes take effect when the server is restarted. If the **onmode** options are set to 0, the memory caches are emptied. The **onstat -g vpcache** option returns information about CPU VP memory block cache statistics.

Related reference:

“scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)” on page 21-101

VPCLASS configuration parameter

Use the VPCLASS configuration parameter to designate a class of virtual processors (VPs), create a user-defined virtual processor, and specify several options.

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

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

onconfig.std values

```
VPCLASS cpu,num=1,noage
```

```
#VPCLASS aio,num=1 (suggested value, but not set)
```

```
#VPCLASS jvp,num=1 (suggested value, but not set)
```

syntax VPCLASS *classname*, *options*

The *classname* variable is required. Additionally, there are several options that you can specify with the VPCLASS configuration parameter.

range of values

Up to 128 bytes of characters. The VPCLASS configuration parameter must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or \$ characters.

takes effect

When the database server is shut down and restarted

utilities

onmode -p (to add or delete VP classes)

refer to

- Specifying user-defined classes of virtual processors, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*
- Specifying a nonyielding user-defined virtual processor (noyield option), in the chapter on virtual processors in the *IBM Informix Administrator's Guide*
- Using **onmode -p** in "onmode -p: Add or remove virtual processors" on page 14-19
- *IBM Informix User-Defined Routines and Data Types Developer's Guide*
- *J/Foundation Developer's Guide*

VPCLASS configuration parameter options and default values

The following table describes the options that you can specify with the VPCLASS configuration parameter. This table also lists the default value and the range of valid values. The options can appear in any order, separated by commas. You cannot use any white space in the options.

Table 1-59. VPCLASS options

Option	Description	Class	Default Value	Range of Values
num	The number of virtual processors that the database server should start initially.	All other classes	1	1 to 10,000
max	The maximum number of virtual processors allowed for this class.	All	Unlimited.	1 to 10,000

Table 1-59. VPCLASS options (continued)

Option	Description	Class	Default Value	Range of Values
aff	The assignment of virtual processors to CPUs if processor affinity is available.	All	Virtual processors are assigned to available processors in round-robin fashion.	Integers from 0 to (number of CPUs -1)
noage	The disabling of priority aging by the operating system, if the operating system implements priority aging.	All	Priority aging is in effect.	noage or omitted
noyield	Creates a class of user-defined virtual processor that is nonyielding.	User defined	Threads will yield.	noyield or omitted

Name of the virtual process class

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

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

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

Some of the predefined names for the virtual processor class have specific default values.

Table 1-60. Default values for some of the predefined names

Name	Description	Class	Default Value	Range of Values
<i>aio,num</i>	The number of asynchronous I/O (AIO) virtual processors that the database server should start initially.	AIO	Not set in <code>onconfig.std</code> . When the aio option is not set, the number of AIO virtual processors is determined by the setting of the <code>AUTO_AIOVPS</code> configuration parameter and is limited to 128: <ul style="list-style-type: none"> If <code>AUTO_AIOVPS</code> is set to 1 (on), the number of AIO virtual processors initially started is equal to the number of AIO chunks. If <code>AUTO_AIOVPS</code> is set to 0 (off), the number of AIO virtual processors started is equal to the greater of 6 or twice the number of AIO chunks. 	1 to 10,000
<i>cpu,num</i>	The number of CPU virtual processors.	CPU	1 if <code>MULTIPROCESSOR</code> is 0, 2 otherwise.	1 to 10,000

Table 1-60. Default values for some of the predefined names (continued)

Name	Description	Class	Default Value	Range of Values
jvp	The number of Java virtual processors. This parameter is required when you use the IBM Informix JDBC Driver. On UNIX, you must define multiple Java virtual processors to run Java user-defined routines in parallel.	JVP	Not set in onconfig.std.	0 to 10,000

For virtual processor classes `tli`, `shm`, `str`, and `soc`, you must set the `VP_class` field for the `NETTYPE` configuration parameter to `NET`.

For example, if the `VPCLASS` configuration parameter is set as:

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

then the `NETTYPE` parameter should be set as follows:

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

For more information on the `NETTYPE` configuration parameter, see “`NETTYPE` Configuration Parameter” on page 1-98.

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

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

Interaction with other configuration parameters

If you use the `VPCLASS` configuration parameter, you must explicitly remove other parameters from your `ONCONFIG` file:

- If you specify the **aio** class name, remove the `NUMAIOVPS` configuration parameter if it exists. This configuration parameter is no longer in use.
- If you specify the **cpu** class name, remove the `NUMCPUVPS`, `AFF_SPROC`, `AFF_NPROCS`, and `NOAGE` configuration parameters if they exist. These configuration parameters are no longer in use.
- If you use a *user-defined* class name, remove the `SINGLE_CPU_VP` configuration parameter.

Creating a user-defined class

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

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

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

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

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

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

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

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

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

Using the num option

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

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

When the database server starts, the number of CPU virtual processors (VPs) is automatically increased to half the number of CPU processors on the database server computer, unless the **SINGLE_CPU_VP** configuration parameter is enabled. The *num* option of the **VPCLASS** configuration parameter for the **cpu** class is not updated when the database server automatically adds CPU VPs. For more information, see "Automatic addition of CPU virtual processors" in the *IBM Informix Performance Guide*.

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

```
num=number
```

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

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

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

Using the max option

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

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

`max=number`

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

Using the aff option

On multiprocessor computers that support *processor affinity*, the affinity option specifies the CPUs to which the database server binds virtual processors. The operating system numbers the CPUs from 0 to (number of CPUs-1).

The affinity option has the following forms, which can be used in combination with each other:

Table 1-61. Various forms of specifying the values for the aff option.

Option form	Examples	Description
<code>aff=(processor_number)</code>	<code>VPCLASS CPU,aff=(3)</code>	The database server binds all of the virtual processors in the class to the CPU that is specified in <code>processor_number</code> . The virtual processor is assigned to CPU 3.
<code>aff=(processor_number, processor_number, processor_number)</code>	<code>VPCLASS CPU,aff=(3,1,5)</code>	The database server binds the virtual processors to the CPUs that are specified in the list. The CPUs can be listed in any order. The virtual processors are assigned to CPUs 3, 1, 5.
<code>aff=(start_range-end_range)</code>	<code>VPCLASS CPU,num=4,aff=(0-3)</code>	The database server binds the virtual processors to the CPUs that are specified in the range. The virtual processors are assigned to CPUs 0, 1, 2, 3.
<code>aff=(start_range-end_range /increment)</code>	<code>VPCLASS CPU,num=4,aff=(1-10/3)</code>	The virtual processors are assigned to every third CPU in the range 1-10, starting with CPU 1. The virtual processors are assigned to CPUs 1, 4, 7, 10.
<code>aff=(processor_number, start_range-end_range, start_range-end_range /increment)</code>	<code>VPCLASS CPU,num=8, aff=(1-10/3,12-14,0)</code>	<p>For the first value 1-10/3, the virtual processors are assigned to every third CPU in the range 1-10, starting with CPU 1. The virtual processors are assigned to CPUs 1, 4, 7, 10.</p> <p>For the second value 12-14, the virtual processors are assigned to 12, 13, and 14.</p> <p>For the third value 0, the virtual processor is assigned to CPU 0.</p>

The database server assigns CPU virtual processors to CPUs in a circular pattern, starting with the first processor number that you specify in the *aff* option. If you specify a larger number of CPU virtual processors than physical CPUs, the database server continues to assign CPU virtual processors starting with the first CPU. For example, suppose you specify the following VPCLASS settings:

```
VPCLASS cpu,num=8,aff=(4-7)
```

The database server makes the following assignments:

- CPU virtual processor number 0 to CPU 4
- CPU virtual processor number 1 to CPU 5
- CPU virtual processor number 2 to CPU 6
- CPU virtual processor number 3 to CPU 7
- CPU virtual processor number 4 to CPU 4
- CPU virtual processor number 5 to CPU 5
- CPU virtual processor number 6 to CPU 6
- CPU virtual processor number 7 to CPU 7

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

Using the **noyield** option

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

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

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

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

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

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

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

Example

If your platform has eight CPUs, your `onconfig` file might include the following VPCLASS configuration parameter entries:

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

Related reference:

“NETTYPE Configuration Parameter” on page 1-98

WSTATS configuration parameter

Use the WSTATS configuration parameter to specify whether the `onstat -g wst` command displays wait statistics for threads within the system.

Attention: You should expect a small performance impact due to the cost of gathering statistical information. Enabling the WSTATS configuration parameter for production systems is not recommended.

onconfig.std *value*

0

range of values

0 = Disable wait statistics

1 = Enable wait statistics

takes effect

When the database server is shut down and restarted

utilities

onstat -g wst

refer to

- “**onstat -g wst** command: Print wait statistics for threads” on page 20-185

You can dynamically change the value of the WSTATS configuration parameter by using the **onmode -wm** or **onmode -wf** command.

USERMAPPING configuration parameter

Use the USERMAPPING configuration parameter to set whether or not the database server accepts connections from mapped users.

Externally authenticated users without operating system (OS) accounts on the Informix host computer can access database server resources when USERMAPPING is turned on by setting the parameter with the BASIC or ADMIN value. The setting of BASIC or ADMIN also determines whether or not mapped users can be granted administrative privileges.

Important: Changing the USERMAPPING configuration parameter from OFF to ADMIN or BASIC is not the only step in setting up Informix for mapped users. You must also enter values in the SYSUSERMAP system catalog table to map users with the appropriate user properties.

default value

OFF

range of values

OFF, BASIC, ADMIN

takes effect

The configuration parameter setting takes effect when the database server is started. Alternatively, you can use the **onmode -wf** command or the **onmode -wm** command to modify the runtime setting of USERMAPPING. If you use the **onmode -wf** command, the setting takes effect when the database server is restarted.

The following table explains USERMAPPING values.

Table 1-62. USERMAPPING Configuration Parameter Settings

Configuration Parameter Setting	Meaning
OFF	Only users that are registered in the Informix host computer OS with a login service can connect to the database server. Externally authenticated users without OS accounts on the Informix host computer cannot connect to database server resources.
BASIC	Users can connect to Informix without an OS account. A user without an OS account cannot perform privileged user operations on the database server, even if the user maps to a server administrator user or group ID.
ADMIN	Users can connect to Informix without an OS account. If a user has authenticated with the identity of a privileged user and is mapped to the proper server administrator group ID, the user can perform DBSA, DBSSO, or AAO work on the database server.

Chapter 2. The sysmaster database

These topics describe the **sysmaster** database and provide reference information for the *system-monitoring interface* (SMI).

These topics include:

- A description of the **sysmaster** database
- Information about how to use SMI tables
- Descriptions of the SMI tables
- A map of the documented SMI tables

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

The sysmaster Database

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

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

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

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

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

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

The buildsmi Script

When you bring the database server up for the first time, it runs a script called **buildsmi**, which is in the **etc** directory. This script builds the database and tables

that support SMI. The database server requires approximately 1750 free pages of logical-log space to build the **sysmaster** database.

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

This script must be run as user **informix** on UNIX, or as a member of the **Informix-Admin** group on Windows, after ensuring that no connections to the **sysmaster** database are made during the build of the database. For example, if a Scheduler task is running when the **buildsmi** script commences, the script fails when the Scheduler attempts to access any of the **sysmaster** tables.

Errors issued while the **buildsmi** script runs are written (on UNIX) to the file **/tmp/buildsmi.out**, or on Windows to the file **%INFORMIXDIR%\etc\buildsmi_out.%INFORMIXSERVER%**, where **%INFORMIXSERVER%** is the name of the Informix instance.

The bldutil.sh Script

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

The System-Monitoring Interface

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

Understanding the SMI Tables

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

The SMI tables provide information about the following topics:

- Auditing
- Checkpoints
- Chunk I/O
- Chunks
- Database-logging status
- Dbspaces
- Disk usage
- Environment variables
- Extents
- Locks

- Networks
- SQL statement cache statistics
- SQL statements
- System profiling
- Tables
- User profiling
- Virtual-processor CPU usage

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

Accessing SMI tables

Any user can use SQL SELECT statements to query an SMI table, but standard users cannot run statements other than the SELECT statement. Users who attempt to run other statements result in permission errors. The administrator can run SQL statements other than SELECT, but the results of such statements are unpredictable.

Tip: For more predictable results, query the views that are associated with each table instead of querying the tables directly.

If you query the **systabpaghdrs** table directly, you must specify an appropriate value for the **pg_partnum** parameter. The value is **pg_partnum > 1048576**. However, if you query the view that is associated with the **systabpaghdrs** table, you do not have to specify this value for the **pg_partnum** parameter.

Informix includes the **sysadinfo** and **sysaudit** tables. Only the user **informix** on UNIX or members of the **Informix-Admin** group on Windows can query the **sysadinfo** and **sysaudit** tables.

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

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

SELECT statements

You can use SELECT statements on SMI tables wherever you can use SELECT against ordinary tables.

For example, you can use SELECT statements ordinary tables from DB-Access, in an SPL routine, with Informix ESQL/C, and so on.

Restriction: You cannot meaningfully reference **rowid** when you query SMI tables. SELECT statements that use **rowid** do not return an error, but the results are unpredictable.

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

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

Triggers and Event Alarms

Triggers based on changes to SMI tables never run. Although you can define triggers on SMI tables, triggers are activated only when an INSERT, UPDATE, or DELETE statement occurs on a table. The updates to the SMI data occur within the

database server, without the use of SQL, so a trigger on an SMI table is never activated, even though the data returned by a SELECT statement indicates that it should be.

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

SPL and SMI Tables

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

Locking and SMI Tables

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

The System-Monitoring Interface Tables

The **sysmaster** database contains many tables that you can use to monitor your system.

Tip: For each system-monitoring interface (SMI) table there is a corresponding view with the same name. For the best results, query the views that are associated with tables instead of querying the underlying tables directly.

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

The database server supports the following SMI tables:

Table	Description	Reference
sysadinfo	Auditing configuration information	"sysadinfo" on page 2-7
sysaudit	Auditing event masks	"sysadinfo" on page 2-7
syscheckpoint	Checkpoint information	"syscheckpoint" on page 2-8
syschkio	Chunk I/O statistics	"syschkio" on page 2-8
syschunks	Chunk information	"syschunks" on page 2-9
syscluster	High-availability cluster information	"syscluster" on page 2-10
syscmsmsla	Connection Manager information	"syscmsmsla" on page 2-12
syscmsmtab	Connection Manager information	"syscmsmtab" on page 2-12
syscompdicts_full	Compression dictionary information	"syscompdicts_full" on page 2-12
sysconfig	Configuration information	"sysconfig" on page 2-14
sysdatabases	Database information	"sysdatabases" on page 2-14

Table	Description	Reference
sysdbslocale	Locale information	"sysdbslocale" on page 2-15
sysdbspaces	Dbpace information	"sysextents" on page 2-17
sysdri	Data-replication information	"sysdri" on page 2-16
sysdual	Is a single-row table	"sysdual" on page 2-17
sysenv	Online server's startup environment	"sysenv" on page 2-17
sysenvses	Session-level environment variable	"sysenvses" on page 2-17
sysextents	Extent-allocation information	"sysextents" on page 2-17
sysextspaces	External spaces information	"sysextspaces" on page 2-18
syssha_lagtime	Secondary server lagtime statistics	"syssha_lagtime Table" on page 2-19
syssha_type	Information about connected servers	"syssha_type" on page 2-20
syssha_workload	Secondary server workload statistics	"syssha_workload" on page 2-20
sysipl	Index page logging information	"sysipl" on page 2-21
syslocks	Active locks information	"syslocks" on page 2-21
syslogs	Logical-log file information	"syslogs" on page 2-22
syslogfil	System log file information	"syslogfil table" on page 2-23
sysmgminfo	Memory Grant Manager/Parallel Data Query information	"sysmgminfo" on page 2-23
sysnetclienttype	Client type network activity	"sysnetclienttype" on page 2-24
sysnetglobal	Global network information	"sysnetglobal" on page 2-24
sysnetworkio	Network I/O	"sysnetworkio table" on page 2-25
sysonlineolog	Online log information	"sysonlineolog" on page 2-25
sysprofile	System-profile information	"sysprofile" on page 2-26
sysproxyagents	information about all the proxy agent threads	"sysproxyagents" on page 2-27
sysproxydistributors	Proxy distributor information	"sysproxydistributors" on page 2-28
sysproxysessions	Information about sessions using redirected-write functionality	"sysproxysessions table" on page 2-28
sysproxytxnops	Information about transactions operations executing via each proxy distributor	"sysproxytxnops table" on page 2-29
sysproxytxns	Information about all of the current transactions executing via each proxy distributor	"sysproxytxns table" on page 2-29
sysptprof	Table information	"sysptprof table" on page 2-30

Table	Description	Reference
sysrepevtreg	Post events to Connection Manager and to the IBM OpenAdmin Tool (OAT) for Informix	"sysrepevtreg table" on page 2-30
sysrepstats	Post events to Connection Manager and to the OAT	"sysrepstats table" on page 2-31
sysrsslog	RS secondary server information	"sysrsslog" on page 2-34
sysscblst	Memory by user	"sysscblst" on page 2-34
sysseprof	Counts of various user actions	"sysseprof" on page 2-35
sysseappinfo	Distributed Relational Database Architecture (DRDA) client-session information.	"sysseappinfo" on page 2-34
syssessions	Description of each user connected	"syssessions" on page 2-35
sysstmx	SMX (server multiplexer group) connection information	"sysstmx" on page 2-37
sysstmkses	SMX (server multiplexer group) session information	"sysstmkses" on page 2-37
syssexplain	SQL statement information enabled by the SET EXPLAIN statement	"syssexplain table" on page 2-37
syssqltrace	SQL statement information	"syssqltrace" on page 2-38
syssqltrace_info	SQL profile trace system information	"syssqltrace_info" on page 2-40
syssqltrace_iter	SQL statement iterators	"syssqltrace_iter" on page 2-40
syssrcrss	RS secondary server statistics	"syssrcrss" on page 2-40
syssrcsds	SD secondary server statistics	"syssrcsds" on page 2-41
systabnames	Database, owner, and table name for the tblspace tblspace	"systabnames" on page 2-41
systabpaghdrs	Page headers	None
systhreads	Wait statistics	"systhreads" on page 2-42
sysstrgrss	RS secondary server statistics	"sysstrgrss" on page 2-42
sysstrgsds	SD secondary server statistics	"sysstrgsds" on page 2-43
sysvpprof	User and system CPU used by each virtual processor	"sysvpprof" on page 2-43

The sysutils Tables

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

Table Description

bar_action

Lists all backup and restore actions that are attempted against an object, except during a cold restore. Use the information in this table to track backup and restore history.

bar_instance

Writes a record to this table for each successful backup. ON-Bar might later use the information for a restore operation.

bar_object

Describes each backup object. This table provides a list of all storage spaces and logical logs from each database server for which at least one backup attempt was made.

bar_server

Lists the database servers in an installation. This table is used to ensure that backup objects are returned to their proper places during a restore.

sysadinfo

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

Column	Type	Description
adtmode	integer	Controls the level of auditing.
adterr	integer	Specifies how the database server behaves when it encounters an error while it writes an audit record.
adtsize	integer	Maximum size of an audit file
adtpath	char(256)	Directory where audit files are written
adtfile	integer	Number of the audit file

sysaudit

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

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

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

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

Column	Type	Description
fail2	integer	Bitmask of the audit mask for failure
fail3	integer	Bitmask of the audit mask for failure
fail4	integer	Bitmask of the audit mask for failure
fail5	integer	Bitmask of the audit mask for failure

syschkio

The **syschkio** system-monitoring interface table provides I/O statistics for individual chunks that the database server manages.

Column	Type	Description
chunknum	smallint	Chunk number
reads	integer	Number of physical reads
pagesread	integer	Number of pages read
writes	integer	Number of physical writes
pageswritten	integer	Number of pages written
mreads	integer	Number of physical reads (mirror)
mpagesread	integer	Number of pages read (mirror)
mwrites	integer	Number of physical writes (mirror)
mpageswritten	integer	Number of pages written (mirror)

syscheckpoint

The **syscheckpoint** table provides information and statistics about checkpoints.

Column	Type	Description
interval	integer	Number of checkpoints since the server was started
type	char(12)	Hard or Interval
caller	char(10)	Caller of the checkpoint
clock_time	integer	Time of day the checkpoint occurred
crit_time	float	Time spent waiting for the critical section to be released
flush_time	float	Time spent flushing pages to disk
cp_time	float	Duration from checkpoint pending until checkpoint done
n_dirty_buffs	integer	Number of dirty buffers
plogs_per_sec	integer	Number of physical log pages processed in a second
llogs_per_sec	integer	Number of logical log pages processed in a second
dskflush_per_sec	integer	Number of buffer pool pages flushed in a second
ckpt_logid	integer	Unique id of the logical log at the checkpoint
ckpt_logpos	integer	Position of the logical log at the checkpoint
physused	integer	Number of pages used in the physical log
logused	integer	Number of pages used in the logical log
n_crit_waits	integer	Number of users who had to wait to enter a critical section

Column	Type	Description
tot_crit_wait	float	Duration spent waiting for all users waiting at the checkpoint critical section block
longest_crit_wait	float	Longest critical section wait
block_time	float	Duration of the checkpoint that blocked the system

syschunks

The **syschunks** table contains a description of each of the chunks that the database server manages.

In the **flags** and **mflags** columns, each bit position represents a separate flag. Thus, it might be easier to read values in the **flags** and **mflags** columns if the values are returned using the HEX function.

Column	Type	Description
chknnum	smallint	Chunk number
dbsnnum	smallint	Dbspace number
nxchknnum	smallint	Number of the next chunk in this dbspace
chksize	integer	Number of pages in this chunk (in units of system default page size)
offset	integer	Page offset of the chunk in its device or path
pagesize	integer	Page size (in bytes)
nfree	integer	<p>Number of free pages in the chunk</p> <p>The free space indicated by the nfree column has different meanings depending on the type of space defined. For dbspaces and smart blobspaces, the free space is reported in units of the basic page size; in blobspaces, the free space is reported in units of the blob space page size.</p> <ul style="list-style-type: none"> • For a dbspace, multiply nfree by the basic page size of either 2KB or 4KB. • For a blobspace, multiply nfree by the blobspace page size. • For a smart blobspace, multiply nfree by the smart blob page size (currently the same as the basic page size).
is_offline	integer	1 If the chunk is offline, 0 if not
is_recovering	integer	1 If the chunk is being recovered, 0 if not
is_blobchunk	integer	1 If the chunk is in a blobspace, 0 if not
is_sbchunk	integer	1 If the chunk is a sbpace, 0 if not
is_inconsistent	integer	1 If the chunk is undergoing logical restore, 0 if not
is_extendable	integer	1 If the chunk is extendable, 0 if not

Column	Type	Description		
flags	smallint	Flags	Hexadecimal	Meaning
		16	0x0010	Chunk is a mirrored chunk
		32	0x0020	Chunk is in offline mode
		64	0x0040	Chunk is in online mode
		128	0x0080	Chunk is in recovery mode
		256	0x0100	Chunk has just been mirrored
		512	0x0200	Chunk is part of a blob space
		1024	0x0400	Chunk is being dropped
		2048	0x0800	Chunk is part of an optical stageblob
		4096	0x1000	Chunk is inconsistent
		8192	0x2000	Chunk is extendable
		16384	0x4000	Chunk was added during roll forward
		32768	0x8000	Chunk has been renamed
		65536	0x10000	Chunk uses big chunk page header
		131072	0x20000	Chunk has a tblspace tblspace extent
		262144	0x40000	No checkpoint has completed since this chunk was initialized (primarily for internal use)
fname	char(256)	Pathname for the file or device of this chunk		
mdsize	integer	Size in pages of the metadata area of a chunk that belongs to a smart blob space. If the chunk does not belong to a smart blob space, the column stores -1.		
mfname	char(256)	Pathname for the file or device of the mirrored chunk, if any		
moffset	integer	Page offset of the mirrored chunk		
mis_offline	integer	1 If mirror is offline, 0 if not		
mis_recovering	integer	1 If mirror is being recovered, 0 if not		
mflags	smallint	Mirrored chunk flags; values and meanings are the same as the flags column.		
udfree	integer	Free space in pages within the user data area of a chunk that belongs to a smart blob space. If the chunk does not belong to a smart blob space, the column stores -1.		
udsize	integer	Size in pages of the user data area of a chunk that belongs to a smart blob space. If the chunk does not belong to a smart blob space, the column stores -1.		

syscluster

The **syscluster** system catalog table stores information about servers in a high-availability cluster. The **syscluster** table has the following columns.

Column	Type	Explanation
name	CHAR(128)	The name of the primary server.
role	CHAR(1)	Code to indicate whether the server is a primary server or secondary server.
syncmode	CHAR(8)	The synchronization mode between the primary server and the secondary server: sync or async.
nodetype	CHAR(8)	The type of server: HDR, RSS, or SDS.
supports_updates	CHAR(1)	Indicates whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the UPDATABLE_SECONDARY configuration parameter).
server_status	CHAR(32)	Indicates the status of the secondary server.
connection_status	CHAR(32)	Indicates the connection status of the secondary server.
delayed_apply	INTEGER	Indicates whether the secondary server waits for a specified amount of time before applying logs (as specified by the DELAY_APPLY configuration parameter).
stop_apply	CHAR(24)	Indicates whether the secondary server has stopped applying log files received from the primary server (as specified by the STOP_APPLY configuration parameter).
logid_sent	INTEGER	Indicates the log ID of the most recent log page sent by the primary server to the secondary server.
logpage_sent	INTEGER	Indicates the page number of the most recent log page sent by the primary server to the secondary server.
logid_acked	INTEGER	Indicates the log ID of the most recent log page the secondary server acknowledged.
logpage_acked	INTEGER	Indicates the page number of the most recent log page the secondary server acknowledged.
ack_time	DATETIME YEAR TO SECOND	Indicates the date and time of the last acknowledged log.
sdscycle	INTEGER	Indicates the cycle number to which the primary server has advanced. Used internally by IBM support to monitor coordination of the primary server with the secondary server.
sdscycle_acked	INTEGER	Indicates the cycle number that the shared disk secondary server has acknowledged. Used internally by IBM support to monitor coordination of the primary server with the secondary server.

syscmsm

The **syscmsm** table is a view of the **syscmsmtab** and **syscmsmsla** tables. It contains Connection Manager service level agreement (SLA) information. The table is updated once every five seconds.

Column	Type	Description
sid	integer	Connection Manager session ID
name	char(128)	Connection Manager name
host	char(256)	Host name
foc	char(128)	Failover configuration (FOC)
flag	integer	Arbitrator flag. A value of 1 indicates that the Connection Manager Arbitrator is active. A value of 0 indicates that the Arbitrator is inactive.
sla_id	integer	Connection Manager service level agreement (SLA) ID
sla_name	char(128)	SLA name
sla_define	char(128)	SLA define
connections	integer	Number of connections made through Connection Manager

syscmsmsla

The **syscmsmsla** table contains Connection Manager service level agreement (SLA) information. The table is updated once every five seconds.

Column	Type	Description
address	int8	CMSLA internal address
sid	integer	Connection Manager session ID
sla_id	integer	Connection Manager service level agreement (SLA) ID
sla_name	char(128)	SLA name
sla_define	char(128)	SLA define
connections	integer	Number of connections made through Connection Manager

syscmsmtab

The **syscmsmtab** table contains Connection Manager information.

Column	Type	Description
address	int8	Connection Manager internal address
sid	integer	Connection Manager session ID
name	char(128)	Connection Manager name
host	char(256)	Host name
foc	char(128)	Failover configuration (FOC)
flag	integer	Arbitrator flag. A value of 1 indicates that the Connection Manager Arbitrator is active. A value of 0 indicates that the Arbitrator is inactive.

syscompdicts_full

The **syscompdicts_full** table and the **syscompdicts** view provide information on all compression dictionaries. The only difference between the table and the view is that, for security purposes, the view does not contain the **dict_dictionary** column.

Only user **informix** can retrieve information from the **syscompdicts_full** table. The **syscompdicts** view is not restricted to user **informix**.

The following table shows the information that the **syscompdicts_full** table and the **syscompdicts** view provide for each compression dictionary.

Table 2-1. Compression Dictionary Information

Column	Type	Description
dict_partnum	integer	Partition number to which the compression dictionary applies
dict_code_version	integer	Version of the code that is creating the compression dictionary 1 is the first version.
dict_dbsnum	integer	Number of the dbspace that the dictionary resides in
dict_create_timestamp	integer	Timestamp that shows when the dictionary was created
dict_create_loguniqueid	integer	Unique ID for the logical log that was created when the dictionary was created
dict_create_logpos	integer	Position within the logical log when the dictionary was created
dict_drop_timestamp	integer	Timestamp that shows when the dictionary was dropped.
dict_drop_loguniqueid	integer	Unique ID for the logical log that was created when the dictionary was dropped.
dict_drop_logpos	integer	Position within the logical log when the dictionary was dropped.
dict_dictionary	byte	Compression dictionary binary object This column is not included in the syscompdicts view.

Sample syscompdicts information

A row of information in the **syscompdicts** view could displays columns containing this information:

```
dict_partnum      1048939
dict_code_version  1
dict_dbsnum       1
dict_create_times+ 1231357656
dict_create_logun+ 11
dict_create_logpos 1695768
dict_drop_timesta+ 0
dict_drop_loguniq+ 0
dict_drop_logpos   0
```

You can use an UNLOAD statement to unload the compression dictionary to a compression dictionary file, as follows:

```
UNLOAD TO 'compression_dictionary_file'
SELECT * FROM sysmaster:syscompdicts_full;
```

sysconfig

The **sysconfig** table describes the effective, original, and default values of the configuration parameters. For more information about the ONCONFIG file and the configuration parameters, see Chapter 1, “Database configuration parameters,” on page 1-1.

Column	Type	Description
cf_id	integer	Unique numeric identifier
cf_name	char(128)	Configuration parameter name
cf_flags	integer	Reserved for future use
cf_original	char(256)	Value in the ONCONFIG file at boot time
cf_effective	char(256)	Value currently in use
cf_default	char(256)	Value provided by the database server if no value is specified in the ONCONFIG file

sysdatabases

The **sysdatabases** view describes each database that the database server manages.

Column	Type	Description
name	char(128)	Database name
partnum	integer	The partition number (tblspace identifier) for the systables table for the database
owner	char(32)	User ID of the creator of the database
created	date	Date created
is_logging	integer	1 If logging is active, 0 if not
is_buff_log	integer	1 If buffered logging, 0 if not
is_ansi	integer	1 If ANSI/ISO-compliant, 0 if not
is_nls	integer	1 If GLS-enabled, 0 if not
is_case_insens	integer	1 If case-insensitive for NCHAR and NVARCHAR columns, 0 if not

Column	Type	Description
flags	smallint	Logging flags (hex values)
		0 No logging
		1 Unbuffered logging
		2 Buffered logging
		4 ANSI/ISO-compliant database
		8 Read-only database
		10 GLS database
		20 Checking of the logging mode of syscdr database bypassed
		100 Changed status to buffered logging
		200 Changed status to unbuffered logging
		400 Changed status to ANSI/ISO compliant
		800 Database logging turned off
		1000 Long ID support enabled

sysdbslocale

The **sysdbslocale** table lists the locale of each database that the database server manages.

Column	Type	Description
db_s_dbsname	char(128)	Database name
db_s_collate	char(32)	The locale of the database

sysdbspaces

The **sysdbspaces** table contains a description of each of the dbspaces that the database server manages.

In the **flags** column, each bit position represents a separate flag. Thus, it might be easier to read values in the **flags** column if the values are returned using the HEX function.

Column	Type	Description
db_snum	smallint	Dbpace number
name	char(128)	Dbpace name
owner	char(32)	User ID of owner of the dbspace
fchunk	smallint	Number of the first chunk in the dbpace
nchunks	smallint	Number of chunks in the dbpace
create_size	decimal	The minimum size of a chunk that can be created for this space using the storage pool.
extend_size	decimal	The minimum size by which a chunk in this storage space can be extended, either manually or automatically.
pagesize	integer	Page size
is_mirrored	integer	1 If dbspace is mirrored, 0 if not

Column	Type	Description		
is_blobspace	integer	1 If the dbspace is a blobspace, 0 if not		
is_sbspace	integer	1 If the dbspace is a sbspace, 0 if not		
is_temp	integer	1 If the dbspace is a temporary dbspace, 0 if not		
flags	smallint	Flags	Hexadecimal	Meaning
		1	0x0001	Dbspace has no mirror
		2	0x0002	Dbspace uses mirroring
		4	0x0004	Dbspace mirroring is disabled
		8	0x0008	Dbspace is newly mirrored
		16	0x0010	Space is a blobspace
		32	0x0020	Blobspace is on removable media
		64	0x0040	Blobspace is on optical media
		128	0x0080	Blobspace has been dropped.
		256	0x0100	Blobspace is an optical stageblob
		512	0x0200	Space is being recovered
		1024	0x0400	Space has been physically recovered
		2048	0x0800	Space is in logical recovery
		32768	0x8000	Space is an sbspace

sysdri

The **sysdri** table provides information about the High-Availability Data-Replication status of the database server.

Column	Type	Description
type	char(50)	High-Availability Data Replication type Possible values: <ul style="list-style-type: none"> • primary • secondary • standard • not initialized
state	char(50)	State of High-Availability Data Replication Possible values: <ul style="list-style-type: none"> • off • on • connecting • failure • read-only
name	char(128)	The name of the other database server in the High-Availability Data-Replication pair
intvl	integer	The High-Availability Data-Replication interval

Column	Type	Description
timeout	integer	The High-Availability Data-Replication timeout value for this database server
lostfound	char(256)	The pathname to the lost-and-found file

sysdual

The **sysdual** table returns exactly one column and one row.

Column	Type	Description
dummy	char(1)	Dummy columns returning "X"

sysenv

The **sysenv** table displays the startup environment settings of the database server.

Column	Type	Description
env_id	integer	Identifier variable number
env_name	char(128)	Environment variable name
env_value	char(512)	Environment variable value

sysenvses

The **sysenvses** table displays the environment variable at the session level.

Column	Type	Description
envses_sid	integer	Session id
envses_id	integer	Identifier variable number
envses_name	char(128)	Session environment variable name
envses_value	char(512)	Session environment variable value

sysextents

The **sysextents** table provides information about extent allocation.

Column	Type	Description
dbname	char(128)	Database name
tablename	char(128)	Table name
chunk	integer	Chunk number
offset	integer	Number of pages into the chunk where the extent begins
size	integer	Size of the extent, in pages

sysextspaces

The **sysextspaces** table provides information about external spaces. Indexes for the **id** column and the **name** column allow only unique values.

Column	Type	Description
id	integer	External space ID
name	char(128)	External space name
owner	char(32)	External space owner
flags	integer	External space flags (reserved for future use)
refcnt	integer	External space reference count.
locsize	integer	Size of external space location, in bytes
location	char (256)	Location of external space

sysfeatures

The **sysfeatures** view provides general information about various features of the Informix database server instance. The **sysfeatures** view is created from an internal table named **syslicenceinfo**, which is stored permanently on the disk. When the database server instances are initialized, the table is pre-allocated with a fixed size which allows tracking of 260 weeks of data. The data wraps every five years.

Metrics are sampled every 15 minutes and only the highest values during the particular week are stored. Each row in the table contains data only for the specific week it represents.

Column	Type	Description
week	smallint	The week that the information was recorded.
year	smallint	The year that the information was recorded.
version	char(12)	The Informix server version.
max_cpu_vps	smallint	The maximum number of CPU virtual processors.
max_vps	smallint	The maximum number of virtual processors.
max_conns	integer	The maximum number of concurrent physical connections on a standalone or high-availability cluster primary server instance.
max_sec_conns	integer	The maximum number of concurrent physical connections on an HDR secondary or RS secondary server instance.
max_sds_clones	smallint	The maximum number of SD secondary server instances connected to the primary server.
max_rss_clones	smallint	The maximum number of RS secondary server instances connected to the primary server.
total_size	integer	The maximum disk space allocated in all chunks (in megabytes).
total_size_used	integer	The maximum disk space used in all chunks (in megabytes).
max_memory	integer	The maximum memory allocated in all segments (in megabytes).

Column	Type	Description
max_memory_used	integer	The maximum memory used in all segments (in megabytes).
is_primary	integer	Indicates whether the server was a primary server in a particular week; 1 = yes, 0 = no.
is_secondary	integer	Indicates whether the server was an HDR secondary server in a particular week; 1 = yes, 0 = no.
is_sds	integer	Indicates whether the server was an SD secondary server in a particular week; 1 = yes, 0 = no (not implemented; always 0).
is_rss	integer	Indicates whether the server was an RS secondary server in a particular week; 1 = yes, 0 = no.
is_er	integer	Indicates whether the server was an enterprise replication server in a particular week; 1 = yes, 0 = no.
is_pdq	integer	Indicates whether the PDQ feature was used on the server instance in the particular week; 1 = yes, 0 = no.

syssha_lagtime Table

The **syssha_lagtime** table provides a history of the amount of time that it took to apply a log record on any of the secondary nodes.

The **syssha_lagtime** table contains a history of the last 20 samplings performed for a particular secondary server.

Column	Type	Description
lt_secondary	CHAR(128)	Name of secondary server
lt_time_last_update	INTEGER	Time at which log record was last updated
lt_lagtime_1	FLOAT	Amount of time required to apply log record for the most recent five-second interval
lt_lagtime_2	FLOAT	Amount of time required to apply log record for the second most recent five-second interval
lt_lagtime_3	FLOAT	Amount of time required to apply log record for the third most recent five-second interval
lt_lagtime_4	FLOAT	Amount of time required to apply log record for the fourth most recent five-second interval
lt_lagtime_5	FLOAT	Amount of time required to apply log record for the fifth most recent five-second interval
lt_lagtime_6	FLOAT	Amount of time required to apply log record for the sixth most recent five-second interval
lt_lagtime_7	FLOAT	Amount of time required to apply log record for the seventh most recent five-second interval
lt_lagtime_8	FLOAT	Amount of time required to apply log record for the eighth most recent five-second interval
lt_lagtime_9	FLOAT	Amount of time required to apply log record for the ninth most recent five-second interval
lt_lagtime_10	FLOAT	Amount of time required to apply log record for the tenth most recent five-second interval
lt_lagtime_11	FLOAT	Amount of time required to apply log record for the eleventh most recent five-second interval

Column	Type	Description
lt_lagtime_12	FLOAT	Amount of time required to apply log record for the twelfth most recent five-second interval
lt_lagtime_13	FLOAT	Amount of time required to apply log record for the thirteenth most recent five-second interval
lt_lagtime_14	FLOAT	Amount of time required to apply log record for the fourteenth most recent five-second interval
lt_lagtime_15	FLOAT	Amount of time required to apply log record for the fifteenth most recent five-second interval
lt_lagtime_16	FLOAT	Amount of time required to apply log record for the sixteenth most recent five-second interval
lt_lagtime_17	FLOAT	Amount of time required to apply log record for the seventeenth most recent five-second interval
lt_lagtime_18	FLOAT	Amount of time required to apply log record for the eighteenth most recent five-second interval
lt_lagtime_19	FLOAT	Amount of time required to apply log record for the nineteenth most recent five-second interval
lt_lagtime_20	FLOAT	Amount of time required to apply log record for the twentieth most recent five-second interval

syssha_type

The **syssha_type** table is a single row table that is used to describe the type of server that is connected.

Column	Type	Description
ha_type	integer	Server type (see table below)
ha_primary	char(128)	Server name (see table below)

Table 2-2. Descriptions for the values in the **syssha_type** table

Value of <i>ha_type</i>	Value of <i>ha_primary</i>	Description
0	NULL	Not part of a high-availability environment
1	<primary server name>	Primary server
2	<primary server name>	HDR secondary server
3	<primary server name>	SD secondary server
4	<primary server name>	RS secondary server

syssha_workload

The **syssha_workload** table contains workload statistics on each of the secondary servers.

Column	Type	Description
wl_secondary	char(128)	Name of secondary server
wl_time_last_update	integer	Time at which workload last updated

Column	Type	Description
wl_type	char(12)	This row contains the ready queue size, user CPU time, and system CPU time
wl_workload_1	float	Most recent workload activity
wl_workload_2	float	Second most recent workload activity
wl_workload_3	float	Third most recent workload activity
wl_workload_4	float	Fourth most recent workload activity
wl_workload_5	float	Fifth most recent workload activity
wl_workload_6	float	Sixth most recent workload activity
wl_workload_7	float	Seventh most recent workload activity
wl_workload_8	float	Eighth most recent workload activity
wl_workload_9	float	Ninth most recent workload activity
wl_workload_10	float	Tenth most recent workload activity
wl_workload_11	float	Eleventh most recent workload activity
wl_workload_12	float	Twelfth most recent workload activity
wl_workload_13	float	Thirteenth most recent workload activity
wl_workload_14	float	Fourteenth most recent workload activity
wl_workload_15	float	Fifteenth most recent workload activity
wl_workload_16	float	Sixteenth most recent workload activity
wl_workload_17	float	Seventeenth most recent workload activity
wl_workload_18	float	Eighteenth most recent workload activity
wl_workload_19	float	Nineteenth most recent workload activity
wl_workload_20	float	Twentieth most recent workload activity

sysipl

The **sysipl** table provides information about the status of index page logging at the primary server.

Column	Type	Description
ipl_status	integer	Index page logging status
ipl_time	integer	Time at which index page logging was enabled

syslocks

The **syslocks** table provides information about all the currently active locks in the database server.

Column	Type	Description
dbname	char(128)	Database name
tablename	char(128)	Table name
rowidlk	integer	Real rowid, if it is an index key lock
keynum	smallint	Key number of index key lock

Column	Type	Description
type	char(4)	Type of lock
		B Byte lock
		IS Intent shared lock
		S Shared lock
		XS Shared key value held by a repeatable reader
		U Update lock
		IX Intent exclusive lock
		SIX Shared intent exclusive lock
		X Exclusive lock
		XR Exclusive key value held by a repeatable reader
owner	integer	Session ID of the lock owner
waiter	integer	Session ID of the user waiting for the lock. If more than one user is waiting, only the first session ID appears.

syslogs

The **syslogs** table provides information about space use in logical-log files. In the **flags** column, each bit position represents a separate flag. For example, for a log file, the **flags** column can have flags set for both current log file and temporary log file. Thus, it might be easier to read values in the **flags** column if the values are returned using the HEX function.

Column	Type	Description
number	smallint	Logical-log file number
uniqid	integer	Log-file ID
size	integer	Number of pages in the log file
used	integer	Number of pages used in the log file
is_used	integer	1 If file is used, 0 if not
is_current	integer	1 If file is the current file, 0 if not
is_backed_up	integer	1 If file has been backed up, 0 if not
is_new	integer	1 If the log has been added since the last level-0 dbspace backup, 0 if not
is_archived	integer	1 If file has been placed on the backup tape, 0 if not
is_temp	integer	1 If the file is flagged as a temporary log file, 0 if not
flags	smallint	Flags Hexadecimal Meaning
		1 0x01 Log file is in use
		2 0x02 File is current log file
		4 0x04 Log file has been backed up
		8 0x08 File is newly added log file
		16 0x10 Log file has been written to dbspace backup media
		32 0x20 Log is a temporary log file

syslogfil table

The syslogfil table provides information about the logical log files.

Table 2-3. Information about the columns in the syslogfil table.

Column	Type	Description
address	int8	Memory address of the logfile structure
number	small integer	Log file number
flags	integer	For a description of the values and their meanings, see the Flag values section below.
fillstamp	integer	Internal timestamp when the log file was filled
filltime	integer	UNIX time when the log file was filled
uniqid	integer	Unique ID for the log file
chunk	integer	Number of the chunk that contains the log file
offset	integer	Page offset in the chunk where log file begins
size	integer	Total number of pages in the log file
used	integer	Number of pages used in the log file

Flag values

The flag values correspond to many of the flag values for the **onstat -l** command.

Hexadecimal	Onstat -l flag value	Meaning
0x1	U	Log file is in use
0x2	C	File is current log file
0x4	B	Log file has been backed up
0x8	A	File is a newly added log file
0x20	None	A temporary log file
0x40	D	Log file will be dropped after the file is archived
0x4000	L	Log file contains the last checkpoint written

sysmgminfo

The **sysmgminfo** table provides an overview of the Memory Grant Manager (MGM) and Parallel Data Query (PDQ) information.

Column	Type	Description
max_query	integer	Maximum number of active queries allowed
total_mem	integer	Total MGM memory
avail_mem	integer	Free MGM memory
total_seq	integer	Total number of sequential scans
avail_seq	integer	Unused sequential scans
active	integer	Number of active MGM queries
ready	integer	Number of ready MGM queries

Column	Type	Description
min_free_mem	integer	Minimum free MGM memory
avg_free_mem	float	Average free MGM memory
std_free_mem	float	Standard free MGM memory
min_free_seq	integer	Minimum free MGM sequential scans
avg_free_seq	float	Average free MGM sequential scans
std_free seq	float	Standard free MGM sequential scans
max_active	integer	Maximum active MGM SQL operations
cnt_active	integer	Number of active MGM SQL operations
avg_active	float	Average active MGM SQL operations
std_active	float	Standard active MGM SQL operations
max_ready	integer	Maximum ready MGM SQL operations
cnt_ready	integer	Number of ready MGM SQL operations
avg_ready	float	Average ready MGM SQL operations
std_ready	float	Standard ready MGM SQL operations

sysnetclienttype

The **sysnetclienttype** table provides an overview of the network activity for each client type.

Column	Type	Description
nc_cons_allowed	integer	Whether or not connections are allowed
nc_accepted	integer	Number of connections that were accepted
nc_rejected	integer	Number of network connections that were rejected
nc_reads	int8	Number of network reads for this client type
nc_writes	int8	Number of network writes for this client type
nc_name	char(18)	Name of the client type

sysnetglobal

The **sysnetglobal** table provides an overview of the system network.

Column	Type	Description
ng_reads	int8	Number of network reads
ng_writes	int8	Number of network writes
ng_connects	int8	Number of network connections
ng_his_read_count	int8	Number of network reads by users who have disconnected ng_his_read_bytes
ng_his_read_bytes	int8	Data transferred to the server by users who have disconnected
ng_his_write_count	int8	Number of network writes by users who have disconnected
ng_his_write_bytes	int8	Data transferred to the client by users who have disconnected

Column	Type	Description
ng_num_netscbs	integer	Number of network subscribers
ng_max_netscbs	integer	Maximum number of network subscribers
ng_free_thres	integer	Threshold for the maximum number of freed buffers in the buffer list
ng_free_cnt	integer	Number of times the ng_free_thres limit has been reached
ng_wait_thres	integer	Threshold for the maximum number of buffers that can be held in the buffer list for one connection
ng_wait_cnt	integer	Number of times the ng_wait_thres limit has been reached
ng_pvt_thres	integer	Threshold for the maximum number of freed buffers in the private buffer queue
ng_netbuf_size	integer	Size of the transport network buffers
ng_buf_alloc	integer	Number of network buffers allocated
ng_buf_alloc_max	integer	Maximum value of allocated network buffers
ng_netscb_id	integer	Next netscb id

sysnetworkio table

The **sysnetworkio** table contains information about the system network.

Column	Type	Description
net_id	integer	Netscb id
sid	integer	Session id
net_netscb	int8	Netscb prt
net_client_type	integer	Client type Int
net_client_name	char(12)	Client protocol name
net_read_cnt	int8	Number of network reads
net_write_cnt	int8	Number of network writes
net_open_time	integer	Time this session connected
net_last_read	integer	Time of the last read from the network
net_last_write	integer	Time of the last write from the network
net_stage	integer	Connect / Disconnect / Receive
net_options	integer	Options from sqlhosts
net_protocol	integer	Protocol
net_type	char(10)	Type of network protocol
net_server_fd	integer	Server fd
net_poll_thread	integer	Poll thread

sysonlinelog

The **sysonlinelog** table provides a view of the information stored in the online.log file.

Column	Type	Description
offset	int8	File offset
next_offset	int8	Offset to the next message
line	char(4096)	Single line of text from the file

sysprofile

The **sysprofile** table contains profile information about the database server.

Column	Type	Description
name	char(13)	Name of profiled event. (See table that follows for a list of possible events.)
value	integer	Value of profiled event. (See table that follows for a list of possible events.)

The following table lists the events that, together with a corresponding value, make up the rows of the **sysprofile** table.

Events Profiled in sysprofile	Description
dskreads	Number of actual reads from disk
bufreads	Number of reads from shared memory
dskwrites	Actual number of writes to disk
bufwrites	Number of writes to shared memory
isamtot	Total number of calls
isopens	isopen calls
isstarts	isstart calls
isreads	isread calls
iswrites	iswrite calls
isrewrites	isrewrite calls
isdeletes	isdelete calls
iscommits	iscommit calls
isrollbacks	isrollback calls
ovlock	Overflow lock table
ovuser	Overflow user table
ovtrans	Overflow transaction table
latchwts	Latch request waits
bufwts	Buffer waits
lockreqs	Lock requests
lockwts	Lock waits
ckptwts	Checkpoint waits
deadlks	Deadlocks
lktouts	Deadlock time-outs
numckpts	Number checkpoints

Events Profiled in sysprofile	Description
plgpagewrites	Physical-log pages written
plgwrites	Physical-log writes
llgrecs	Logical-log records
llgpagewrites	Logical-log writes
llgwrites	Logical-log pages written
pagreads	Page reads
pagwrites	Page writes
flushes	Buffer-pool flushes
compress	Page compresses
fgwrites	Foreground writes
lruwrites	Least-recently used (LRU) writes
chunkwrites	Writes during a checkpoint
btradata	Read-ahead data pages read through index leaf node
btraidx	Read-ahead data pages read through index branch or root node
dpra	Data pages read into memory with read-ahead feature
rapgs_used	Read-ahead data pages that user used
seqscans	Sequential scans
totalsorts	Total sorts
memsorts	Sorts that fit in memory
disksorts	Sorts that did not fit in memory
maxsortspace	Maximum disk space used by a sort

sysproxyagents

The **sysproxyagents** table contains information about all proxy agent threads. Proxy agent threads run on the primary server and accept requests from secondary servers to process DML operations. The primary server also contains a proxy distributor that handles secondary server updates. Secondary servers determine how many instances of the proxy distributor to create based on the `UPDATABLE_SECONDARY` setting in the secondary server's `ONCONFIG` file.

Column	Type	Description
tid	integer	Transaction ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the secondary server session.
flags	integer	Flags of the proxy agent thread.
proxy_id	integer	ID of the proxy distributor on behalf of the currently executing proxy agent thread (TID).
source_session_id	integer	ID of the user's session on the secondary server.
proxy_txn_id	integer	Number of the current transaction. These numbers are unique to the proxy distributor.
current_seq	integer	The sequence number of the current operation in the current transaction.

Column	Type	Description
sqlerrno	integer	Error number of any SQL error (or 0 on success)
iserrno	integer	Error number of any ISAM/RSAM error (or 0 on success)

sysproxydistributors

The **sysproxydistributors** table contains information about the proxy distributors.

On the primary server, this table contains information about all of the proxy distributors in a high-availability cluster. On a secondary server, this table contains information about only those proxy distributors that are assigned to process updates to the secondary server.

Column	Type	Description
node_name	char	Name of the secondary server as it is known by the primary server (for example, INFORMIXSERVER, HA_ALIAS, and so on).
proxy_id	integer	ID of the proxy distributor. These IDs are unique within a high-availability cluster.
transaction_count	integer	Number of transactions currently being processed by the proxy distributor.
hot_row_total	integer	Total number of hot rows ever handled by the proxy distributor. A hot row is a row on a secondary server that is updated multiple times by more than one client. When a row is updated multiple times, the secondary server reads the before image from the primary server by placing an update lock on the row if the most recent update operation from a different session is not replayed on the secondary server.

sysproxysessions table

The **sysproxysessions** table contains information about each of the sessions that are using redirected-write functionality. This table is only valid on the secondary server.

The following table provides information about the columns in the **sysproxysessions** table:

Column	Type	Description
session_id	integer	ID of a user's session on the secondary server.
proxy_id	integer	ID of the proxy distributor on behalf of which the proxy agent thread (TID) is running
proxy_tid	integer	Transaction ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the secondary server session.
proxy_txn_id	integer	Number of the current transaction. These numbers are unique to the proxy distributor.
current_seq	integer	The sequence number of the current operation in the current transaction.

Column	Type	Description
pending_ops	integer	The number of operations buffered on the secondary server that have not yet been sent to the primary server.
reference_count	integer	Indicates the number of threads (for example, sqlxexec, sync reply, recovery, and so on) that are using the information for this transaction. When reference_count equals 0, the transaction processing has completed (either successfully or unsuccessfully).

sysproxytxnops table

The **sysproxytxnops** table contains information about each of the transactions that are running through each proxy distributor.

On the primary server, this table contains information about all of the proxy distributors in the high-availability cluster. On a secondary server, this table only contains information about the proxy distributors used to process updates to the secondary server.

The following table provides information about the columns in the **sysproxytxnops** table:

Column	Type	Description
proxy_id	integer	ID of the proxy distributor. These IDs are unique within a high-availability cluster.
proxy_txn_id	integer	Number of the transaction. These numbers are unique to the proxy distributor.
sequence_number	integer	The number of the operation.
operation_type	char(10)	The type of operation to be performed; Insert, Update, Delete, or Other.
rowidn	integer	The ID of the row on which to apply the operation.
table	char	The full table name, trimmed to fit a reasonable length. Format: <i>database:owner.tablename</i>
sqlerrno	integer	Error number of any SQL error (or 0 on success)

sysproxytxns table

The **sysproxytxns** table contains information about all of the current transactions that are running through each proxy distributor.

On the primary server, this table contains information about each of the proxy distributors in the high-availability cluster. On a secondary server, this table only contains information about the proxy distributors used to process updates to the secondary server.

The following table provides information about the columns in the **sysproxytxns** table:

Column	Type	Description
proxy_id	integer	ID of the proxy distributor. These IDs are unique within a high-availability cluster.

Column	Type	Description
proxy_txn_id	integer	Number of the transaction. These numbers are unique to the proxy distributor.
reference_count	integer	Indicates the number of threads (for example, sqlexec, sync reply, recovery, and so on) that are using the information for this transaction. When the count becomes 0 this indicates the transaction processing is complete. (either successfully or unsuccessfully).
pending_ops	integer	On the primary server: the number of operations received from the secondary server that have not yet been processed. On the secondary server, the number of operations buffered on the secondary server that have not yet been sent to the primary server.
proxy_sid	integer	Proxy Session ID

sysptprof table

The **sysptprof** table lists information about a tblspace. Tblspaces correspond to tables.

Profile information for a table is available only when a table is open. When the last user who has a table open closes it, the tblspace in shared memory is freed, and any profile statistics are lost.

The following table provides information about the columns in the **sysptprof** table:

Column	Type	Description
dbname	char(128)	Database name
tablename	char(128)	Table name
partnum	integer	Partition (tblspace) number
lockreqs	integer	Number of lock requests
lockwts	integer	Number of lock waits
deadlks	integer	Number of deadlocks
lktouts	integer	Number of lock timeouts
isreads	integer	Number of isreads
iswrites	integer	Number of iswrites
isrewrites	integer	Number of isrewrites
isdeletes	integer	Number of isdeletes
bufreads	integer	Number of buffer reads
bufwrites	integer	Number of buffer writes
seqscans	integer	Number of sequential scans
pagreads	integer	Number of page reads
pagwrites	integer	Number of page writes

sysrepevtreg table

Use the **sysrepevtreg** pseudo table to register for a pre-defined set of events from the Connection Manager, the IBM OpenAdmin Tool (OAT) for Informix, or any

client application. After registering events through the **sysrepevtreg** pseudo table, Connection Manager, OAT, or any client application can receive event data by querying the table.

The following table provides information about the columns in the **sysrepevtreg** table:

Column	Type	Description
evt_bitmap	integer	Event ID bitmap
evt_timeout	integer	Maximum time in seconds that the client can wait for event data. Valid timeout values are: <ul style="list-style-type: none">• 0; no wait (default)• -1; wait forever• <i>n</i> (where <i>n</i> > 0) wait <i>n</i> seconds
evt_hwm	integer	Pending event list high-water mark
evt_info	char(256)	Event information (Not yet implemented)

sysrepstats table

Use the **sysrepstats** table to post events to Connection Manager and to the IBM OpenAdmin Tool (OAT) for Informix. Connection Manager, OAT, and client applications can communicate with each other by posting events to the **sysrepstats** pseudo table.

The following table provides information about the columns in the **sysrepstats** table:

Column	Type	Description
repstats_type	integer	Event ID
repstats_subtype	integer	Sub event ID
repstats_time	integer	Time at which event was initiated
repstats_ver	integer	Version number of event data
repstats_desc	lvarchar	Event data

User Interface for sysrepstats and sysrepevtreg Tables

Client applications can post events to Connection Manager or to other clients by inserting event information into the **sysrepstats** pseudo table. Client applications can register events using the sysmaster pseudo table **sysrepevtreg**, and receive event data by issuing select or fetch statements against the **sysrepstats** pseudo table.

Posting events to the **sysrepstats** pseudo table provides the ability for programs such as the IBM OpenAdmin Tool (OAT) for Informix to communicate with Connection Manager. By posting events to the **sysrepstats**, you can issue control messages to Connection Manager without having to directly connect to Connection Manager itself.

When Connection Manager registers that it wishes to receive events, it passes a bitmap of the event types that it wants to receive. As events are received, they are posted to the thread that placed the request.

Event Classes

The following table lists each event class, its bit value, and a description of the event class.

Event class name	Bit value	Description
REPEVT_CLUST_CHG	0x1	Event class for High-Availability cluster changes
REPEVT_CLUST_PERFSTAT	0x2	Event class for workload statistics for the server nodes in a High-Availability cluster
REPEVT_CLUST_LATSTAT	0x4	Event class for replication latency information for server nodes in a High-Availability cluster
REPEVT_CM_ADM	0x8	Connection Manager administration commands
REPEVT_SRV_ADM	0x10	Event class for server mode changes
REPEVT_ER_ADM	0x20	Event class for events related to Enterprise Replication (ER)
REPEVT_CLIENT	0x40	User-defined client event

Sub-events for the Event Class REPEVT_CLUST_CHG

The following table lists sub-events for the event class REPEVT_CLUST_CHG:

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_CLUST_ADD	1	Adding new node to a High-Availability cluster	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_DROP	2	Dropping a node from a High-Availability cluster	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_CON	3	High-Availability secondary node connected to primary server	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_DIS	4	High-Availability secondary node disconnected from primary server	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_NEWPRIM	5	High-Availability primary node changed	Only at secondary servers in a High-Availability cluster
REPEVT_SUB_CLUST_DROFF	6	HDR secondary node disconnected from primary server	HDR primary and secondary servers
REPEVT_SUB_CLUST_DRON	7	HDR secondary node connected to primary server	HDR primary and secondary servers

Sub-events for the Event Class REPEVT_CLUST_PERFSTAT

The following table lists sub-events for the event class REPEVT_CLUST_PERFSTAT:

Sub-event name	Value	Description	Event available at:
REPEVT_ SUB_LOCAL_PERFSTAT	1	Work load statistics for local server	All servers in a High-Availability cluster
REPEVT_ SUB_REMOTE_PERFSTAT	2	Work load statistics for High-Availability secondary servers	Only at the primary server in a High-Availability cluster

Sub-events for the Event Class REPEVT_CLUST_LATSTAT

The following table lists sub-events for the event class REPEVT_CLUST_LATSTAT:

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_LOCAL_LATSTAT	1	Replication latency statistics for secondary servers in a High-Availability cluster	Only at the primary server in a High-Availability cluster

Sub-events for the Event Class REPEVT_CM_ADM

The following table lists sub-events for the event class REPEVT_CM_ADM:

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_CM_ADM_REQ	1	Command request	All Informix server instances
REPEVT_SUB_CM_ADM_ACK	2	Command response	All Informix server instances
REPEVT_SUB_CM_REG	3	Connection Manager registered with server	All Informix server instances
REPEVT_SUB_CM_DEREG	4	Connection Manager de-registered with server	All Informix server instances
REPEVT_SUB_CM_FATAL	5	Connection Manager terminated without de-registering with server	All Informix server instances

Sub-events for the Event Class REPEVT_SRV_ADM

The following table lists sub-events for the event class REPEVT_SRV_ADM:

Sub-event name	Value	Description	Event available at:
REPEVT_ SUB_SRV_BLK	1	Server blocked due to DDRBLOCK	All Informix server instances

Sub-event name	Value	Description	Event available at:
REPEVT_ SUB_SRV_UBLK	2	Server unblocked; DDRBLOCK removed	All Informix server instances

Sub-events for the Event Class REPEVT_ER_ADM

The following table lists sub-events for the event class REPEVT_ER_ADM:

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_ER_SPOOL_FULL	1	ER blocked while waiting for space to be added in either the queue data sbspace or dbspace, or in the grouper paging sbspace.	Enterprise Replication server nodes

sysrsslog

The **sysrsslog** table captures information about RS secondary servers at the primary server.

Column	Type	Description
server_name	char(128)	Server name
from_cache	integer	Total pages read from log buffer cache
from_disk	integer	Total pages read from disk
logpages_tossed	integer	Total number of log pages not written to log buffer cache

sysscblst

These columns of the **sysscblst** table provide information about session memory amounts.

Column	Type	Description
memtotal	integer	Total memory available
memused	integer	Total memory used

sys sesappinfo

The **sys sesappinfo** table in the **sysmaster** displays information on Distributed Relational Database Architecture (DRDA) client sessions. The **sys sesappinfo** table has the following columns.

Column	Type	Explanation
sesapp_sid	INTEGER	Client session ID
sesapp_name	CHAR(128)	Session application name
sesapp_value	CHAR(512)	Session value

sysstesprof

The **sysstesprof** table lists cumulative counts of the number of occurrences of user actions such as writes, deletes, or commits.

Column	Type	Description
sid	integer	Session ID
lockreqs	integer	Number of locks requested
locksheld	integer	Number of locks currently held
lockwts	integer	Number of times waited for a lock
deadlks	integer	Number of deadlocks detected
lktouts	smallint	Number of deadlock time-outs
logrecs	integer	Number of logical-log records written
isreads	integer	Number of reads
iswrites	integer	Number of writes
isrewrites	integer	Number of rewrites
isdeletes	integer	Number of deletes
iscommits	integer	Number of commits
isrollbacks	integer	Number of rollbacks
longtxs	integer	Number of long transactions
bufreads	integer	Number of buffer reads
bufwrites	integer	Number of buffer writes
seqscans	integer	Number of sequential scans
pagreads	integer	Number of page reads
pagwrites	integer	Number of page writes
total_sorts	integer	Number of total sorts
dsksorts	integer	Number of sorts that did not fit in memory
max_sortdiskspace	integer	Maximum space used by a sort
logspused	integer	Number of bytes of logical-log space used by current transaction of session
maxlogsp	integer	Maximum number of bytes of logical-log usage for any single transaction since the session started

syssessions

The **syssessions** table provides general information on each user connected to the database server. In the **state** column, each bit position represents a separate flag. Thus, it might be easier to read values in the **state** column if the values are returned using the HEX function.

Column	Type	Description
sid	integer	Session ID
username	char(32)	User ID
uid	smallint	User ID number
pid	integer	Process ID of the client

Column	Type	Description		
hostname	char(16)	Hostname of client		
tty	char(16)	Name of the user's stderr file		
connected	integer	Time that user connected to the database server		
feprogram	char(255)	Absolute path of the executable program or application		
pooladdr	integer	Session pool address		
is_wlatch	integer	1 if the primary thread for the session is waiting for a latch		
is_wlock	integer	1 if the primary thread for the session is waiting for a lock		
is_wbuff	integer	1 if the primary thread for the session is waiting for a buffer		
is_wckpt	integer	1 if the primary thread for the session is waiting for a checkpoint		
is_wlogbuf	integer	1 if the primary thread for the session is waiting for a log buffer		
is_wtrans	integer	1 if the primary thread for the session is waiting for a transaction		
is_monitor	integer	1 if the session is a special monitoring process		
is_incrit	integer	1 if the primary thread for the session is in a critical section		
state	integer	Flags	Hexadecimal	Meaning
		1	0x00000001	User structure in use
		2	0x00000002	Waiting for a latch
		4	0x00000004	Waiting for a lock
		8	0x00000008	Waiting for a buffer
		16	0x00000010	Waiting for a checkpoint
		32	0x00000020	In a read call
		64	0x00000040	Writing logical-log file to backup tape
		128	0x00000080	ON-Monitor (UNIX)
		256	0x00000100	In a critical section
		512	0x00000200	Special daemon
		1024	0x00000400	Archiving
		2048	0x00000800	Clean up dead processes
		4096	0x00001000	Waiting for write of log buffer
		8192	0x00002000	Special buffer-flushing thread
		16384	0x00004000	Remote database server
		32768	0x00008000	Deadlock timeout used to set RS_timeout
		65536	0x00010000	Regular lock timeout
		262144	0x00040000	Waiting for a transaction
		524288	0x00080000	Primary thread for a session
		1048576	0x00100000	Thread for building indexes
		2097152	0x00200000	B-tree cleaner thread

sysismx

The **sysismx** table provides SMX (server multiplexer group) connection information.

Column	Type	Description
address	int8	SMX pipe address
name	char(128)	Target server name
encryption_status	char(20)	Enabled or disabled
buffers_sent	integer	Number of buffers sent
buffers_rcv	integer	Number of buffers received
bytes_sent	int8	Number of bytes sent
bytes_rcv	int8	Number of bytes received
reads	integer	Number of read calls
writes	integer	Number of write calls
retries	integer	Number of write call retries

sysismxes

The **sysismxes** table provides SMX (server multiplexer group) session information.

Column	Type	Description
name	char(128)	Target server name
address	int8	SMX session address
client_type	char(20)	SMX client type
reads	integer	Number of read calls
writes	integer	Number of write calls

sysiqexplain table

The **sysiqexplain** pseudo table stores information about SQL queries if recording this information is enabled by the SET EXPLAIN statement. The information stored includes the plan of the query optimizer, an estimate of the number of rows returned, and the relative cost of the query.

Table 2-4. The sysiqexplain pseudo table

Column	Type	Description
sqx_sessionid	INTEGER	The session ID associated with the SQL statement.
sqx_sdbno	INTEGER	The position of the query in the array of session IDs.
sqx_iscurrent	CHAR	Whether the query is the current SQL statement.
sqx_executions	INTEGER	The total number of executions of the query.
sqx_cumtime	FLOAT	The cumulative time to run the query.
sqx_bufreads	INTEGER	The number of buffer reads performed while running the query.
sqx_pagereads	INTEGER	The number of page reads performed while running the query.

Table 2-4. The `syssexplain` pseudo table (continued)

Column	Type	Description
<code>sqx_bufwrites</code>	INTEGER	The number of buffer writes performed while running the query.
<code>sqx_pagewrites</code>	INTEGER	The number of page writes performed while running the query.
<code>sqx_tot sorts</code>	INTEGER	The total number of sorts performed while running the query.
<code>sqx_dsk sorts</code>	INTEGER	The number of disk sorts performed while running the query.
<code>sqx_sort spmax</code>	INTEGER	The maximum disk space required by a sort.
<code>sqx_conbno</code>	SMALLINT	The position in the conblock list.
<code>sqx_ismain</code>	CHAR	Whether the query is in the main block for the statement.
<code>sqx_selflag</code>	VARCHAR(200,0)	The type of SQL statement, for example: SELECT, UPDATE, DELETE.
<code>sqx_estcost</code>	INTEGER	The estimated cost of the query.
<code>sqx_estrows</code>	INTEGER	The estimated number of rows returned by the query.
<code>sqx_seqscan</code>	SMALLINT	The number of sequential scans used by the query.
<code>sqx_srtscan</code>	SMALLINT	The number of sort scans used by the query.
<code>sqx_autoindex</code>	SMALLINT	The number of autoindex scans used by the query.
<code>sqx_index</code>	SMALLINT	The number of index paths used by the query.
<code>sqx_remsql</code>	SMALLINT	The number of remote paths used by the query.
<code>sqx_mrgjoin</code>	SMALLINT	The number of sort-merge joins used by the query.
<code>sqx_dynhashjoin</code>	SMALLINT	The number of dynamic hash joins used by the query.
<code>sqx_keyonly</code>	SMALLINT	The number of key-only scans used by the query.
<code>sqx_tempfile</code>	SMALLINT	The number of temporary files used by the query.
<code>sqx_tempview</code>	SMALLINT	The number of temporary tables for views created by the query.
<code>sqx_secthreads</code>	SMALLINT	The number of secondary threads used by the query.
<code>sqx_sqlstatement</code>	CHAR	The SQL query that was run.

sysqltrace

The `sysqltrace` table provides detailed information about a single SQL statement.

Column	Type	Description
<code>sql_id</code>	int8	Unique SQL execution ID
<code>sql_address</code>	int8	Address of the statement in the code block
<code>sql_sid</code>	int	Database session ID of the user running the SQL statement
<code>sql_uid</code>	int	User ID of the statement running the SQL
<code>sql_stmttype</code>	int	Statement type

Column	Type	Description
sql_stmtname	char(40)	Statement type displayed as a word
sql_finishtime	int	Time this statement completed (UNIX)
sql_begintxtime	int	Time this transaction started
sql_runtime	float	Statement execution time
sql_pgreads	int	Number of disk reads for this SQL statement
sql_bfreads	int	Number of buffer reads for this SQL statement
sql_rdcache	float	Percentage of time the page was read from the buffer pool
sql_bfidxreads	int	Number of index page buffer reads
sql_pgwrites	int	Number of pages written to disk
sql_bfwrites	int	Number of pages modified and returned to the buffer pool
sql_wrcache	float	Percentage of time a page was written to the buffer pool but not to disk
sql_lockreq	int	Total number of locks required by this SQL statement
sql_lockwaits	int	Number of times the SQL statement waited on locks
sql_lockwttime	float	Time the system waited for locks during SQL statement
sql_logspace	int	Amount of space the SQL statement used in the logical log
sql_sorttotal	int	Number of sorts that ran for the statement
sql_sortdisk	int	Number of sorts that ran on disk
sql_sortmem	int	Number of sorts that ran in memory
sql_executions	int	Number of times the SQL statement ran
sql_totalltime	float	Total amount of time spent running the statement
sql_avgtime	float	Average amount of time spent running the statement
sql_maxtime	float	Maximum amount of time spent executing the SQL statement
sql_numioawaits	int	Number of times an I/O operation had to wait
sql_avgioawaits	float	Average amount of time that the SQL statement had to wait
sql_totalioawaits	float	Total amount of time that the SQL statement had to wait for I/O. This excludes any asynchronous I/O.
sql_rowspersec	float	Average number of rows (per second) produced
sql_estcost	int	Cost associated with the SQL statement
sql_estrows	int	Estimated number of rows returned for the SQL statement as predicted by the optimizer
sql_actualrows	int	Number of rows returned for the SQL statement
sql_sqlerror	int	SQL error number
sql_isamerror	int	RSAM/ISAM error number
sql_isollevel	int	Isolation level of the SQL statement.
sql_sqlmemory	int	Number of bytes needed to execute the SQL statement
sql_numiterators	int	Number of iterators used by the statement
sql_database	char(128)	Database name

Column	Type	Description
sql_numtables	int	Number of tables used in executing the SQL statement
sql_tablelist	char(4096)	List of table names directly referenced in the SQL statement. If the SQL statement fires triggers that execute statements against other tables, the other tables are not listed.
sql_statement	char(1600)	SQL statement that ran

syssqltrace_info

The **syssqltrace_info** table describes information about the SQL profile trace system.

Column	Type	Description
flags	integer	SQL trace flags
ntraces	integer	Number of items to trace
tracesize	integer	Size of the text to store for each SQL trace item
duration	integer	Trace buffer (in seconds)
sqlseen	int8	Number of SQL items traced since start or resizing
starttime	integer	Time tracing was enabled
memoryused	int8	Number of bytes of memory used by SQL tracing

syssqltrace_iter

The **syssqltrace_iter** table lists the SQL statement iterators.

Column	Type	Description
sql_id	int8	SQL execution ID
sql_address	int8	Address of the SQL statement block
sql_itr_address	int8	Address of the iterator
sql_itr_id	int	Iterator ID
sql_itr_left	int	Iterator ID to the left
sql_itr_right	int	Iterator ID to the right
sql_itr_cost	int	Iterator cost
sql_itr_estrows	int	Iterator estimated rows
sql_itr_numrows	int	Iterator actual rows processed
sql_itr_type	int	Iterator type
sql_itr_misc	int	Iterator miscellaneous flags
sql_it_info	char(256)	Iterator miscellaneous flags displayed as text

syssrcrss

The **syssrcrss** table provides RS secondary server related statistics at the primary server.

Column	Type	Description
address	int8	RS secondary server control block address
server_name	char(128)	Database server name
server_status	char(20)	Quiescent, active, or inactive
connection_status	char(20)	Connected or disconnected
log_transmission_status	char(20)	Active or blocked
next_page_tosend_log_uniq	integer	Unique log ID of next page to send
next_page_tosend_log_page	integer	Page number of next page to send
seq_tosend	integer	Sequence ID of last buffer sent
last_seq_acked	integer	Sequence ID of last buffer acknowledged

syssrcsds

The **syssrcsds** table provides SD secondary server related statistics at the primary server.

The **syssrcsds** table contains these columns:

Column	Type	Description
address	int8	SD secondary server control block address
source_server	char(128)	Primary database server name
connection_status	char(20)	Connected or disconnected
last_received_log_uniq	integer	Unique log ID of last log page received
last_received_log_page	integer	Page number of last log page received
next_lpgtoread_log_uniq	integer	Unique log ID of next log page to read
next_lpgtoread_log_page	integer	Page number of next log page to read
last_acked_lsn_uniq	integer	Unique log ID of last LSN acknowledged
last_acked_lsn_pos	integer	Log position of last LSN acknowledged
last_seq_received	integer	Sequence ID of last buffer received
last_seq_acked	integer	Sequence ID of last buffer acknowledged
cur_pagingfile	char(640)	Current paging file name
cur_pagingfile_size	int8	Current paging file size
old_pagingfile	char(640)	Old paging file name
old_pagingfile_size	int8	Old paging file size

systabnames

The **systabnames** table describes each table that the database server manages.

Column	Type	Description
partnum	integer	tblspace identifier
dbname	char(128)	Database name
owner	char(32)	User ID of owner
tablename	char(128)	Table name

Column	Type	Description
collate	char(32)	Collation associated with a database that supports GLS

systhreads

The **systhreads** table provides information about each thread.

Column	Type	Description
th_id	INTEGER	The numeric identifier of the thread.
th_addr	INTEGER	The memory address of the thread control block.
th_joinlist	INTEGER	If a list of the threads are waiting for this thread to exit, the th_joinlist column shows the address of the first thread in the list.
th_joinnext	INTEGER	If a list of the threads are waiting for this thread to exit, the th_joinnext column shows the address of the next thread in the join list.
th_joinnee	INTEGER	The address of the thread whose exit this thread is waiting for.
th_name	CHAR(12)	The name of the thread.
th_state	INTEGER	The status code of the thread.
th_priority	INTEGER	The priority of the thread.
th_class	INTEGER	The code for the class of virtual processor that thread will run on.
th_vpid	INTEGER	The ID of the virtual processor that the thread was last scheduled to run on.
th_mtxwait	INTEGER	The address of the mutex that this thread is waiting for.
th_conwait	INTEGER	The address of the condition that this thread is waiting for.
th_waketime	INTEGER	The time of the expiration of the last sleep. The time is calculated by an internal clock. A value of -1 means that the time value is indeterminate.
th_startwait	INTEGER	The time when the last wait began. The time is calculated by an internal clock.
th_startrun	INTEGER	The time when the last execution began. The time is calculated by an internal clock.

systmgrss

The **systmgrss** table provides RS secondary server related statistics at the RS secondary server.

Column	Type	Description
address	int8	RS secondary server control block address
source_server	char(128)	Source server serving the RS secondary server
connection_status	char(20)	Connected or disconnected
last_received_log_uniq	integer	Unique log ID of last log page received
last_received_log_page	integer	Page number of last log page received
last_seq_received	integer	Sequence ID of last buffer received

Column	Type	Description
last_seq_acked	integer	Sequence ID of last buffer acknowledged

systrgsds

The **systrgsds** table provides SD secondary server related statistics at the SD secondary server.

The **systrgsds** table contains these columns:

Column	Type	Description
address	int8	SD secondary server control block address
source_server	char(128)	Source server serving the SD secondary server
connection_status	char(20)	Connected or disconnected
last_received_log_uniq	integer	Unique log ID of last log page received
last_received_log_page	integer	Page number of last log page received
next_lptoread_log_uniq	integer	Unique log ID of next log page to read
next_lptoread_log_page	integer	Page number of next log page to read
last_acked_lsn_uniq	integer	Unique log ID of last LSN acknowledged
last_acked_lsn_pos	integer	Log position of last LSN acknowledged
last_seq_received	integer	Sequence ID of last buffer received
last_seq_acked	integer	Sequence ID of last buffer acknowledged
cur_pagingfile	char(640)	Current paging file name
cur_pagingfile_size	int8	Current paging file size
old_pagingfile	char(640)	Old paging file name
old_pagingfile_size	int8	Old paging file size

sysvpprof

The **sysvpprof** table lists user and system CPU time for each virtual processor.

Column	Type	Description
vpid	integer	Virtual processor ID
	char(50)	Type of virtual processor: <ul style="list-style-type: none"> • cpu • adm • lio • pio • aio • tli • soc • str • shm • opt • msc • adt

Column	Type	Description
usercpu	float	Number of microseconds of user time
syscpu	float	Number of microseconds of system time

The SMI Tables Map

Figure 2-1 displays the columns in some of the SMI tables.

sysadinfo	sysaudit	syschkio	syschunks	sysconfig	sysdatabases
adtmode	username	chunknum	chknum	cf_id	name
adterr	succ1	reads	dbsnum	cf_name	partnum
adtsize	succ2	pagesread	nxchknum	cf_flags	owner
adtpath	succ3	writes	chksize	cf_originals	created
adtfile	succ4	pageswritten	offset	cf_effective	is_logging
	succ5	mreads	nfree	cf_default	is_buff_log
	fail1	mpagesread	ls_offline		is_ansi
	fail2	mwrites	is_recovering		is_nls
	fail3	mpageswritten	is_blobchunk		flags
	fail4		is_sbchunk		
	fail5		is_inconsistent		
			flags		
			fname		
			mfname		
			moffset		
			mis_offline		
			mis_recovering		
			mflags		

sysdbslocale	sysdbspaces	sysdri	sysextents	sysextspaces	syslocks
dbs_dbname	dbsnum	type	dbsname	id	dbsname
dbs_collate	name	state	tablename	name	tablename
	owner	name	chunk	owner	rowidlk
	fchunk	intvl	offset	flags	keynum
	nchunks	timeout	size	refcnt	type
	is_mirrored	lostfound		loctype	owner
	is_blobspace			location	waiter
	is_sbpace				
	is_temp				
	flags				

Figure 2-1. Columns in the SMI tables

syslogs	sysprofile	sysptprof	sysesprof	sysessions
number	name	dbsname	sid	sid
uniqid	value	tabname	lockreqs	username
size		partnum	locksheld	uid
used		lockreqs	lockwts	pid
is_used		lockwts	deadlks	hostname
is_current		deadlks	lktouts	tty
is_backed_up		lktouts	logrecs	connected
is_new		isreads	isreads	feprogram
is_archived		iswrites	iswrites	pooladdr
is_temp		isrewrites	isrewrites	is_wlatch
flags		isdeletes	isdeletes	is_wlock
		bufreads	iscommits	is_wbuff
		bufwrites	isrollbacks	is_wckpt
		seqscans	longtxs	is_wlogbuf
		pagreads	bufreads	is_wtrans
		pagwrites	bufwrites	is_monitor
			seqscans	is_incrit
			pagreads	state
			pagwrites	
			total_sorts	
			dsksorts	
			max_sort diskpace	
			logspused	
			maxlogsp	

syseswts	systabnames	sysvpprof
sid	partnum	vpid
reason	dbsname	class
numwaits	owner	usercpu
cumtime	tabname	syscpu
maxtime	collate	

Information from onstat in the SMI Tables

To obtain information provided by the **onstat** utility, you can use SQL to query appropriate SMI tables. The following table indicates which SMI tables to query to obtain the information provided by a given **onstat** option. For descriptions of the **onstat** options, see “Monitor the database server status” on page 20-18.

onstat Option	SMI Tables to Query	onstat Fields <i>Not</i> in SMI Tables
-d	sysdbspaces syschunks	address bpages
-D	sysdbspaces syschkio	
-F	sysprofile	address flusher snoozer state data
-g ath	systhreads	

onstat Option	SMI Tables to Query	onstat Fields <i>Not</i> in SMI Tables
-g dri	sysdri	Last DR CKPT (id/pg)
-g glo	sysvpprof	Listing of virtual processors by
-g ipl	sysipl	
-g rss	sysrsslog systgrss syssrcrs	
-g his	sysqltracing	
-g sds	syssrcsds systrgsds	
-g smx	sysmx	
-g smx ses	sysmxses	
-k	syslocks	address lklist tblsnum
-l	syslogs sysprofile	All physical-log fields (except numpages and numwrits) All logical-log buffer fields (except numrecs, numpages, and numwrits) address begin % used
-p	sysprofile	
-u	syssessions syssexprof	address wait nreads nwrites

Chapter 3. The sysadmin Database

The **sysadmin** database contains the tables that contain and organize the Scheduler tasks and sensors, store data collected by sensors, and record the results of Scheduler jobs and SQL administration API functions.

By default, only user **informix** is granted access to the **sysadmin** database; other users can be granted access to the **sysadmin** database.

Do not drop or alter the **sysadmin** database because it is used by several important database server components. You can, however, move the **sysadmin** database from its default root dbspace location if the root dbspace does not have enough space for storing task properties and command history information. To move the **sysadmin** database, use the SQL administration API **admin()** or **task()** function with the **reset sysadmin** argument.

Important: Moving the **sysadmin** database resets the database back to the original state when it was first created; all data, command history, and results tables are lost. Only built-in tasks, sensor, and thresholds remain in the **sysadmin** tables.

Related reference:

“reset sysadmin argument: Move the sysadmin database (SQL administration API)”
on page 21-99

The Scheduler Tables

The Scheduler stores information about tasks and sensors in five tables in the **sysadmin** database: **ph_task**, **ph_run**, **ph_group**, **ph_alert**, and **ph_threshold**.

The Scheduler is an administrative tool that enables the database server to execute database functions and procedures at predefined times or as determined internally by the server. The five tables used by the Scheduler contain built-in tasks and sensors that run automatically. You can also add your own tasks and sensors by inserting rows into these tables. These tables have relationships between their columns that are described in the following illustration.

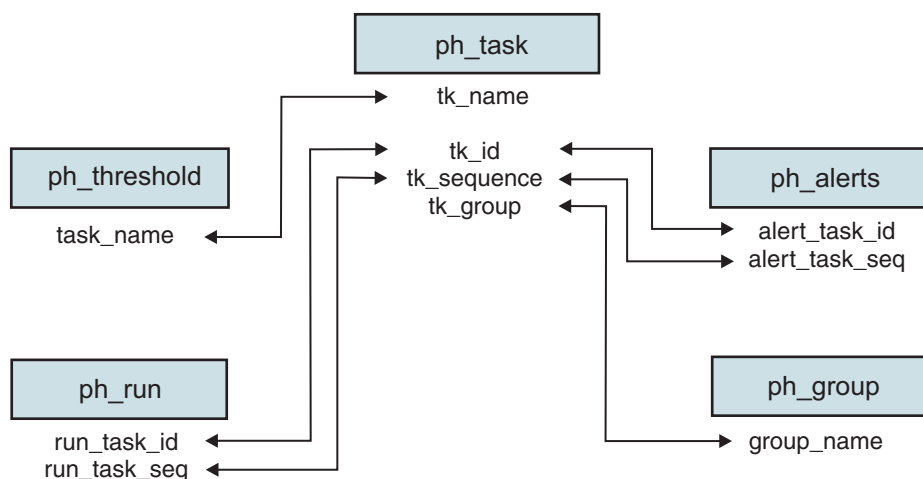



Figure 3-1. Relationships between the Scheduler tables

For detailed information about using the Scheduler, see the *IBM Informix Administrator's Guide*.

Related concepts:

 Scheduler tables (Administrator's Guide)

The **ph_task** Table

The **ph_task** table contains information about Scheduler tasks and sensors. The **ph_task** table contains built-in tasks and sensors that are scheduled to run automatically.

*Table 3-1. The **ph_task** table*

Column	Type	Description
tk_id	serial	Sequential job ID. System updated; do not modify. Referenced in the alert_task_id column in the ph_alert table and in the run_task_id column in the ph_run table.
tk_name	char(36)	Job name. A unique index on this column ensures that no two names are the same. Referenced in the task_name column of the ph_threshold table.
tk_description	lvvarchar	Description about what the task or sensor does.
tk_type	char(18)	Type of job: <ul style="list-style-type: none">• TASK: Invokes an action at a specific time and frequency• SENSOR: (Default) A task that collects, stores, and purges data to or from a result table• STARTUP TASK: A task that runs only when the server starts• STARTUP SENSOR: A sensor that runs only when the server starts
tk_sequence	integer	Current data collection number. System updated; do not modify. Referenced in the alert_task_id column of the ph_alert table and the run_task_seq column of the ph_run table.
tk_result_table	varchar	Results table name for storing data collected by a sensor. The table is created by the CREATE TABLE statement in the tk_create column.
tk_create	lvvarchar	The CREATE TABLE statement used to create the results table to store data collected by a sensor. One of the columns in the table must be named ID and hold the tk_sequence value. This value indicates the age of the row and can be used for purging the row.

Table 3-1. The *ph_task* table (continued)

Column	Type	Description
tk_dbs	varchar(250)	The database in which the task is run. Default is sysadmin.
tk_execute	lvarchar	The SQL statement to execute.
tk_delete	interval day(2) to second	Data older than this interval is deleted from the result table. Default is 1:00:00 (one day).
tk_start_time	datetime hour to second	Time when the task or sensor starts. Default is 08:00:00.
tk_stop_time	datetime hour to second	Time of day after which the task or sensor cannot be scheduled to be run. The database server schedules the next execution on the next valid day. Default is 19:00:00. Can be NULL, indicating no stop time.
tk_frequency	interval day(2) to second	How often this task or sensor runs. Default is 1 (once a day).
tk_next_execution	datetime year to second	Next time this task or sensor will be run. After a startup task or sensor has run, this value is NULL. When a task or a sensor is enabled, the database server calculates this time from the values of tk_start_time , tk_stop_time , and tk_frequency columns, and the days of the week the task or sensor is enabled, according to the values of tk_monday , tk_tuesday , tk_wednesday , tk_thursday , tk_friday , tk_saturday , tk_sunday columns.
tk_total_executions	integer	The number of times that the task or sensor was run. System updated; do not modify. Default is 0.
tk_total_time	float	Total time spent executing this task or sensor. System updated; do not modify Default is 0.0 seconds.
tk_monday	boolean	Whether the task or sensor is run on Monday. Default is T (true).
tk_tuesday	boolean	Whether the task or sensor is run on Tuesday. Default is T (true).
tk_wednesday	boolean	Whether the task or sensor is run on Wednesday. Default is T (true).

Table 3-1. The *ph_task* table (continued)

Column	Type	Description
tk_thursday	boolean	Whether the task or sensor is run on Thursday. Default is T (true).
tk_friday	boolean	Whether the task or sensor is run on Friday. Default is T (true).
tk_saturday	boolean	Whether the task or sensor is run on Saturday. Default is T (true).
tk_sunday	boolean	Whether the task or sensor is run on Sunday. Default is T (true).
tk_attributes	integer	Flags System updated; do not modify.
tk_group	varchar(128)	Group name. Must be the same as a value in the group_name column in the ph_group table. Default is MISC.
tk_enable	boolean	Whether the task or sensor is enabled. Default is T (the task is enabled).
tk_priority	integer	Job priority, on a scale of 0- 5, with higher numbers indicating higher priority. If there are several jobs to execute simultaneously, the job with the highest priority executes first. Default is 0 (low priority).

The **ph_run** Table

The **ph_run** table contains information about how and when each Scheduler task or sensor ran.

Table 3-2. The *ph_run* table

Column	Type	Description
run_id	serial	Sequential ID generated during execution. System updated; do not modify.
run_task_id	integer	The job ID. Referenced from the tk_id column of the ph_task table.
run_task_seq	integer	Unique sequence number of the task or sensor. Referenced from the tk_sequence column of the ph_task table.
run_retcode	integer	The return code from a stored procedure function or a user-defined routine, or the SQLCode from SQL statements.

Table 3-2. The *ph_run* table (continued)

Column	Type	Description
run_time	datetime year to second	When this task or sensor was run.
run_duration	float	The time that it took to run this task or sensor (in seconds).
run_ztime	integer	The time when the server statistics (the onstat -z command) were last run.
run_btime	integer	The time when the server was started.
run_mtttime	integer	Internal counter of the server.

The *ph_group* Table

The **ph_group** table contains information about the Scheduler group names. The **ph_group** table contains several groups that are used to categorize built-in tasks and sensors, as well as the default group MISC.

Table 3-3. The *ph_group* table

Column	Type	Description
group_id	serial	Group ID. System updated; do not modify.
group_name	varchar(128)	Unique name of the group. Referenced in the tk_group column of the ph_task table.
group_description	lvarchar	Description of the group.

The *ph_alert* Table

The **ph_alert** table contains information about event alarms generated by the database server or alerts generated by the Scheduler. Alerts that are associated with built-in tasks and sensors are automatically added to the **ph_alert** table.

Table 3-4. The *ph_alert* table

Column	Type	Description
id	serial	The alert ID. System generated; do not modify.
alert_task_id	serial	The task or sensor ID. Must be the same as a value in the tk_id column in the ph_task table. The task ID for event alarms is 15.
alert_task_seq	integer	Identifies which invocation of a task created the alert. System generated; do not modify. Referenced from the tk_sequence column in the ph_task table.

Table 3-4. The *ph_alert* table (continued)

Column	Type	Description
alert_type	char(8)	<p>The type of alert:</p> <ul style="list-style-type: none"> • INFO (Default) • WARNING • ERROR <p>The severity of an alert or event alarm is indicated by the combination of the alert type and the alert color. See Table 3-5 on page 3-7.</p>
alert_color	char(15)	<p>The color of the alert:</p> <ul style="list-style-type: none"> • GREEN (Default) • YELLOW • RED <p>The severity of an alert or event alarm is indicated by the combination of the alert type and the alert color. See Table 3-5 on page 3-7.</p>
alert_time	datetime year to second	<p>The time when the alert was generated.</p> <p>System updated; do not modify.</p>
alert_state	char(15)	<p>Indicates which state the object is in:</p> <p>NEW (Default) The alert was added and no other action has occurred on this alert.</p> <p>IGNORED The alert was acknowledged by the DBA and no action was taken.</p> <p>ACKNOWLEDGED The alert was acknowledged by the DBA.</p> <p>ADDRESSED The alert was addressed by the DBA.</p>
alert_state_changed	datetime year to second	<p>The last time that the state was changed.</p> <p>System updated; do not modify.</p>
alert_object_type	char(15)	<p>The type of object that the alert is for:</p> <ul style="list-style-type: none"> • ALARM • CHUNK • DATABASE • DBSPACE • INDEX • MISC (Default) • SERVER • SQL_STATEMENT • TABLE • USER
alert_object_name	varchar(255)	<p>The name of the object that the alert is for or the event alarm class ID.</p>
alert_message	lvarchar	<p>The detailed message describing the alert or event alarm.</p>

Table 3-4. The *ph_alert* table (continued)

Column	Type	Description
alert_action_dbs	lvvarchar(256)	The name of the database to use for the corrective action. Default is sysadmin .
alert_action	lvvarchar	The corrective action. An SQL script to invoke to correct the problem. This script must comply with all multi-statement prepare rules. Can be NULL if no action is available.
alert_object_info	bigint	For alerts of type ALARM, the event ID of the event alarm.

The following table defines the alert colors for the three types of messages and event alarms.

Table 3-5. Alert types and colors

Message Type	Green	Yellow	Red
Informative	A status message indicating a component's operation status. An event alarm of severity 1 (not noteworthy).	An important status message. An event alarm of severity 2 (information).	A status message that requires action.
Warning	A warning from the database that was automatically addressed.	A future event that needs to be addressed. An event alarm of severity 3 (attention).	A predicted failure is imminent. Immediate action is required.
Error	A failure in a component corrected itself.	A failure in a component corrected itself but might need DBA action.	A failure in a component requires DBA action. An event alarm of severity 4 (emergency) or 5 (fatal).

The **ph_alerts** view shows alert information and associated task or sensor information.

Related concepts:

“Events in the *ph_alert* Table” on page C-3

The **ph_threshold** Table

The **ph_threshold** table contains information about thresholds for Scheduler tasks.

The **ph_threshold** table contains built-in thresholds that are associated with built-in tasks and sensors. For example, a threshold named COMMAND HISTORY RETENTION determines the length of time rows should remain in the **command_history** table.

Table 3-6. The *ph_threshold* table

Column	Type	Description
id	integer	Threshold ID.
name	char	The name of the threshold.
task_name	varchar	Scheduler task name associated with the threshold. Must be the same as a value in the tk_name column in the ph_task table.
value	lvarchar	The value of the threshold.
value_type	char	The data type of the value column: <ul style="list-style-type: none">• STRING• NUMERIC• NUMERIC(p,s)
description	lvarchar	A long description of the threshold.

Results Tables

Results tables contain historical data about sensors that are run by the Scheduler.

Most sensors create a new table to store their results. The name of the table is listed in the **tk_result_table** column in the **ph_task** table. The structure of the table is defined by the CREATE TABLE statement in the **tk_create** column of the **ph_task** table.

The built-in sensors automatically create results tables when they run that start with the prefix **mon_**.

Table 3-7. Results tables

Column	Type	Description
ID	integer	The iteration sequence number of the sensor. Must be set to \$DATA_SEQ_ID. Referenced from the run_task_seq column of the ph_run table.
<i>user columns</i>	any	You can specify any types of columns to hold the information returned by a sensor.

The command_history table

The **command_history** table contains the list and results of all the SQL administration API functions that were run in the previous 30 days.

The **command_history** table shows each SQL administration API function that was run and displays information about the user who ran the function, the time the function was run, the primary arguments of the function, and the message returned when the database server completed running the function.

Table 3-8. The `command_history` table

Column	Data Type	Description
<code>cmd_number</code>	serial	The unique ID for each row.
<code>cmd_exec_time</code>	datetime year-to-second	The time that the function started.
<code>cmd_user</code>	varchar	The user who ran the function.
<code>cmd_hostname</code>	varchar	The name of the host computer from which the function was run.
<code>cmd_executed</code>	varchar	The primary argument of the function that was run.
<code>cmd_ret_status</code>	integer	Return code.
<code>cmd_ret_msg</code>	lvarchar	Return message.

The following table shows sample arguments and the associated results messages in a `command_history` table.

Table 3-9. Example information in a `command_history` table

Argument (<code>cmd_executed</code>)	Message Returned (<code>cmd_ret_msg</code>)
set sql tracing on	SQL tracing on with 1000 buffers of 2024 bytes.
create dbspace	Space 'space12' added.
checkpoint	Checkpoint completed.
add log	Added 3 logical logs to dbspace logdbs.

To display the command history, run the following SQL statement from the `sysadmin` database:

```
SELECT * FROM command_history;
```

The size of the `command_history` table

Depending on how many SQL administration API functions are run, the `command_history` table can grow quite large. You can change the amount of time that information is retained in the `command_history` table by updating the `value` field of the `COMMAND HISTORY RETENTION` row in the `ph_threshold` table.

You can also use SQL statements like `DELETE` or `TRUNCATE TABLE` to manually remove the data from this table.

Related tasks:

 Viewing SQL administration API history (Administrator's Guide)

The `storagepool` table

The `storagepool` table in `sysadmin` database contains information about all of the entries in the storage pool in an Informix instance. Each entry represents free space that the server can use when automatically expanding a storage space.

Table 3-10. The `storagepool` table

Column	Type	Description
<code>entry_id</code>	SERIAL	The ID of the storage pool entry.

Table 3-10. The **storagepool** table (continued)

Column	Type	Description
path	VARCHAR (255)	The path for the file, directory, or device that the server can use when additional storage space is required.
beg_offset	BIGINT	The initial offset in kilobytes into the device at which the server can begin allocating space. If the storage pool information is for a directory, the end offset value is 0.
end_offset	BIGINT	The initial offset in kilobytes into the device at which the server must stop allocating space. If the storage pool information is for a directory, the offset value is 0.
chunk_size	BIGINT	The initial size of a chunk allocated from this entry.
status	VARCHAR (255)	The status of the storage pool entry. Status values are: Active = A functional storage pool entry. The server can allocate chunks from this entry. Full = There is no more free space in the storage pool entry. The server cannot allocate any more chunks from this entry. Error = The storage pool entry generated an error when the server tried to allocate a chunk from the entry.
priority	INT	The priority of the directory, file, or device when the server searches through the storage pool for space. The server allocates space from a high-priority entry before it allocates space from a lower priority entry. 1 = High priority 2 = Medium priority 3 = Low priority
last_alloc	DATETIME (year to second)	The date and time of the last allocation from this entry.
logid	INT	The ID of the log that was current at the time this entry was last used. The server uses this flag with the logused value when choosing between entries of identical priorities.
logused	INT	The position within the log that was current at the time this entry was last used. The server uses this flag with the logid value when choosing between entries of identical priorities.

Chapter 4. Disk Structures and Storage

In This Chapter

The database server achieves its high performance by managing its own I/O. The database server manages storage, search, and retrieval. As the database server stores data, it creates the structures it needs to search for and retrieve the data later. The database server disk structures also store and track control information needed to manage logging and backups. Database server structures contain all the information needed to ensure data consistency, both physical and logical.

Before you read this chapter, familiarize yourself with the disk-space terms and definitions in the chapter on where data is stored in the *IBM Informix Administrator's Guide*.

This chapter discusses the following topics related to disk data structures:

- Dbspace structure and storage
- Storage of simple large objects
- Sbspace structure
- Time stamps
- Database and table creation: what happens on disk

Dbspace Structure and Storage

This section explores the disk structures and storage techniques that the database server uses to store data in a dbspace.

Related concepts:

 Forest of trees indexes (Performance Guide)

Structure of the Root Dbspace

As part of disk-space initialization, the database server initializes specific structures in the initial chunk of the root dbspace.

The following structures are initialized:

- Twelve reserved pages
- The first chunk free-list page
- The tblspace tblspace
- The physical log
- The logical-log files
- The database tblspace

The ROOTNAME, ROOTOFFSET, ROOTPATH, and ROOTSIZE configuration parameters specify the size and location of the initial chunk of the root dbspace. If the root dbspace is mirrored, the MIRROROFFSET and MIRRORPATH configuration parameters specify the mirror-chunk location. For more information about these parameters, see Chapter 1, "Database configuration parameters," on page 1-1.

To see the structure of the root chunk use the **oncheck -pe** command. For more information, see “oncheck -ce, -pe: Check the chunk-free list” on page 8-10.

Reserved Pages

The first 12 pages of the initial chunk of the root dbspace are reserved pages. Each reserved page contains specific control and tracking information used by the database server.

To obtain a listing of the contents of your reserved pages, execute the command **oncheck -pr**. To also list information about the physical-log and logical-log pages, including the active physical-log pages, run the **oncheck -pR** command.

The following example shows sample **oncheck -pr** output for interval checkpoints:

```
Time of checkpoint      10/25/2005 17:05:20
Checkpoint Interval    1234
```

The database server also stores current configuration information in a reserved page called PAGE_CONFIG. If you change the configuration parameters from the command line and run the **oncheck -pr** command without shutting down and restarting the database server, the configuration values in the command output do not match the current values in the reserved pages. The **oncheck** utility returns a warning message.

The following example shows sample output of the contents of a PAGE_CONFIG reserved page.

```
...
Validating Informix database server reserved pages - PAGE_CONFIG
ROOTNAME                rootdbs
ROOTPATH                /home/dyn_srv/root_chunk
ROOTOFFSET              4
ROOTSIZE                8000
MIRROR                 0
MIRRORPATH
MIRROROFFSET            0
PHYSFILE               1000
LOGFILES                5
LOGSIZE                500
MSGPATH                /home/dyn_srv/online.log
CONSOLE                 /dev/tty5
...
...
```

Structure of a Regular Dbspace

After disk-space initialization, you can add new dbspaces. When you create a dbspace, you assign at least one chunk (either raw or cooked disk space) to the dbspace. This chunk is referred to as the initial chunk of the dbspace. Figure 4-1 on page 4-3 illustrates the structure of the initial chunk of a regular (nonroot) dbspace.

When the dbspace is first created, it contains the following structures:

- Two reserved pages
- The first chunk free-list page in the chunk
- The tblspace **tblspace** for this dbspace
- Unused pages

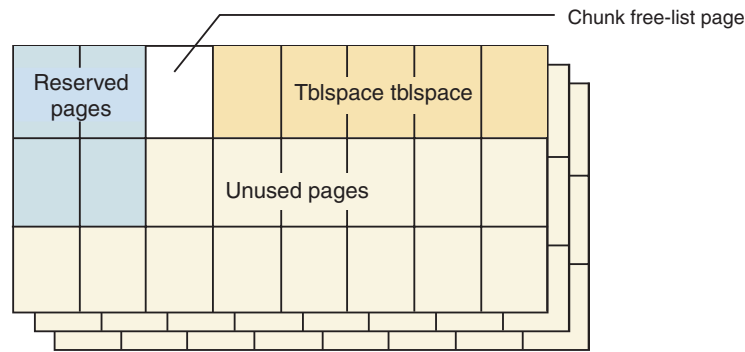


Figure 4-1. Initial Chunk of Regular Dbspace

Structure of an Additional Dbspace Chunk

You can create a dbspace that contains more than one chunk. The initial chunk in a dbspace contains the tblspace **tblspace** for the dbspace. Additional chunks do not. When an additional chunk is first created, it contains the following structures:

- Two reserved pages
- The first chunk free-list page
- Unused pages

Figure 4-2 illustrates the structure of all additional chunks in a dbspace. (The structure also applies to additional chunks in the root dbspace.)

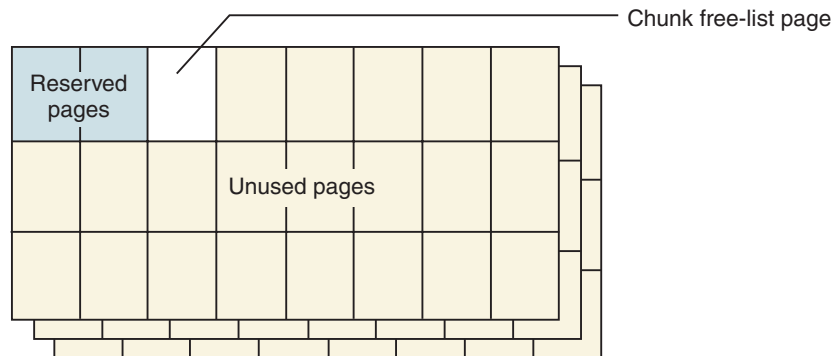


Figure 4-2. Additional Dbspace Chunk

Structure of a Mirror Chunk

Each mirror chunk must be the same size as its primary chunk. When a mirror chunk is created, the database server writes the contents of the primary chunk to the mirror chunk immediately.

The mirror chunk contains the same control structures as the primary chunk. Mirrors of blobspace, sbpace, or dbspace chunks contain the same physical contents as their primary counterpart after the database server brings them online.

Figure 4-3 on page 4-4 illustrates the mirror-chunk structure as it appears after the chunk is created.

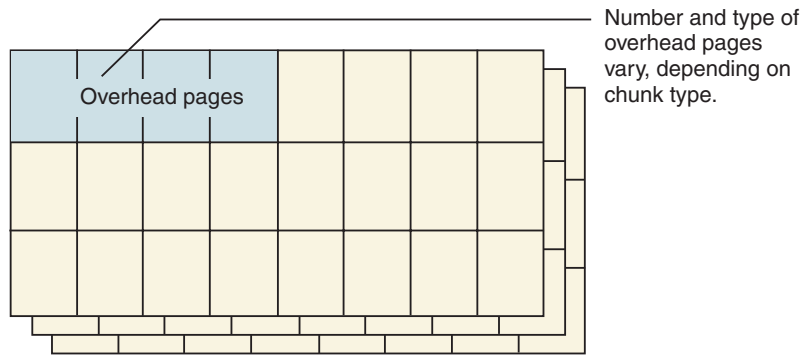


Figure 4-3. Mirror-Chunk Structure

The mirror-chunk structure always shows no free space because all of its space is reserved for mirroring. For more information, see the chapter on what is mirroring in the *IBM Informix Administrator's Guide*.

Structure of the Chunk Free-List Page

In every chunk, the page that follows the last reserved page is the first of one or more chunk free-list pages that tracks available space in the chunk. For a non-root chunk, the initial length of the free space is equal to the size of the chunk minus three pages. If an additional chunk free-list page is needed to accommodate new entries, a new chunk free-list page is created in one of the free pages in the chunk. Figure 4-4 illustrates the location of the free-list page.

Use **oncheck -pe** to obtain the physical layout of pages in the chunk. For more information, see “oncheck -ce, -pe: Check the chunk-free list” on page 8-10.

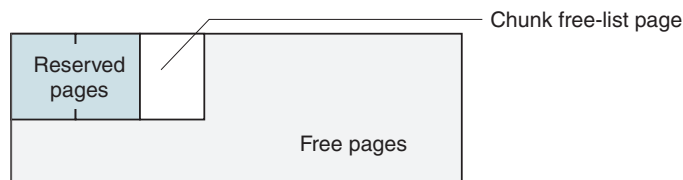


Figure 4-4. Free-List Page

Structure of the Tblspace Tblspace

Each dbspace contains a tblspace called the *tblspace tblspace* that describes all tblspaces in the dbspace. When the database server creates a tblspace, it places an entry in the *tblspace tblspace* that describes the characteristics of the newly created tblspace. You cannot drop or move a chunk containing a *tblspace tblspace*.

A dbspace can have a maximum number of 2^{20} tblspaces.

The default size of the first and next extents depends on whether the dbspace is the root dbspace or not, as shown in the following table.

Table 4-1. Default sizes for each extent and type of dbspace

Type of dbspace	Default Size of First Extent	Default Size of Next Extents
Root	<ul style="list-style-type: none"> • 500 KB for a 2 kilobyte page system • 1000 KB for a 4 kilobyte page system 	<ul style="list-style-type: none"> • 100 KB for a 2 kilobyte page system • 200 KB for a 4 kilobyte page system
Non-root	<ul style="list-style-type: none"> • 100 KB for a 2 kilobyte page system • 200 KB for a 4 kilobyte page system 	<ul style="list-style-type: none"> • 100 KB for a 2 kilobyte page system • 200 KB for a 4 kilobyte page system

You can specify a non-default size for the first and next extents for a **tblspace** in the following ways:

- For the root dbspace, set the **TBLTBLFIRST** and **TBLTBLNEXT** configuration parameters.
- For non-root dbspaces, use the **onspaces** utility **-ef** and **-en** options when you create a dbspace.

Tblspace Tblspace Entries

To display information on the **tblspace**, use the **oncheck -pt** command. For more information, see “**oncheck -pt** and **-pT**: Display **tblspaces** for a Table or Fragment” on page 8-19.

Component	Description
Page header	24 bytes, standard page-header information
Page-ending time stamp	4 bytes
Tblspace header	68 bytes, general tblspace information
Column information	Each special column in the table is tracked with an 12-byte entry. (A special column is defined as a VARCHAR, BYTE, or TEXT data type.)
Tblspace name	80 bytes, <i>database.owner.tblspace</i>
Index information	Each index on the table contains a 20-byte header that contains general information about the index, followed by a 4-byte entry for each column component of the index
Extent information	Each extent allocated to this tblspace is tracked with a 12-byte entry

Tblspace Numbers

Each **tblspace** that is described in the **tblspace** receives a **tblspace** number. This **tblspace** number is the same value that is stored as the **partnum** field in the **sysables** system catalog table and as the **partn** field in the **sysfragments** system catalog table.

The following SQL query retrieves the **partnum** for every table in the database (these can be located in several different dbspaces) and displays it with the table name and the hexadecimal representation of **partnum**:

```
SELECT tabname, tabid, partnum, HEX(partnum) hex_tblspace_name FROM sysables
```

If the output includes a row with a table name but a **partnum** of 0, this table consists of two or more table fragments, each located in its own tblspace. For example, Figure 4-5 shows a table called **account** that has **partnum** 0.

tablename	tabid	partnum	hex_tblspace_name
sysfragments	25	1048611	0x00100023
branch	100	1048612	0x00100024
teller	101	1048613	0x00100025
account	102	0	0x00000000
history	103	1048615	0x00100027
results	104	1048616	0x00100028

Figure 4-5. Output from *systables* Query with *partnum* Values

To obtain the actual tblspace numbers for the fragments that make up the table, you must query the **sysfragments** table for the same database. Figure 4-6 shows that the **account** table from Figure 4-5 has three table fragments and three index fragments.

tabid	fragtype	partn	hex_tblspace_name
102	T	1048614	0x00100026
102	T	2097154	0x00200002
102	T	3145730	0x00300002
102	I	1048617	0x00100029
102	I	2097155	0x00200003
102	I	3145731	0x00300003

Figure 4-6. Output from *sysfragments* Table with *partn* Values

Tblspace Number Elements

The first page in a tblspace is logical page 0. (Physical page numbers refer to the address of the page in the chunk.) The root space tblspace **tblspace** is always contained in the first dbspace and on logical page 1 within the tblspace **tblspace**. (The bitmap page is page 0.)

Tblspace Tblspace Size

These tblspace **tblspace** pages are allocated as an extent when the dbspace is initialized. If the database server attempts to create a table, but the tblspace **tblspace** is full, the database server allocates a next extent to the tblspace.

When a table is removed from the dbspace, its corresponding entry in the tblspace **tblspace** is deleted.

Tblspace Tblspace Bitmap Page

The first page of the tblspace **tblspace**, like the first page of any initial extent, is a bitmap that describes the page fullness of the following pages. Each page that follows has an entry on the bitmap page. If needed, additional bitmap pages are located throughout the contiguous space allocated for the tblspace, arranged so that each bitmap describes only the pages that follow it, until the next bitmap or the end of the dbspace. Bitmap pages fall at distinct intervals within tblspaces pages. Each bitmap page describes a fixed number of pages that follow it.

Structure of the Database Tblspace

The database tblspace appears only in the initial chunk of the root dbspace. The database tblspace contains one entry for each database managed by the database server. Figure 4-7 illustrates the location of the database tblspace.

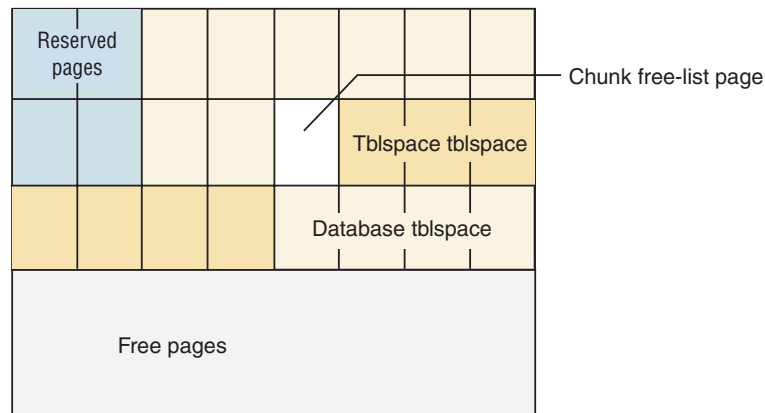


Figure 4-7. Database Tblspace Location in Initial Chunk of Root Dbspace

Database Tblspace Number

The tblspace number of the database tblspace is always 0x100002. This tblspace number appears in an **onstat -t** listing if the database tblspace is active.

Database Tblspace Entries

Each database tblspace entry includes the following five components:

- Database name
- Database owner
- Date and time that the database was created
- The tblspace number of the **systables** system catalog table for this database
- Flags that indicate logging mode

The database tblspace includes a unique index on the database name to ensure that every database is uniquely named. For any database, the **systables** table describes each permanent table in the database. Therefore, the database tblspace only points to the detailed database information located elsewhere.

When the root dbspace is initialized, the database tblspace first extent is allocated. The initial-extent size and the next-extent size for the database tblspace are four pages. You cannot modify these values.

Structure and Allocation of an Extent

This section covers the following topics:

- Extent structure
- Next-extent allocation

Extent Structure

An extent is a collection of contiguous pages within a dbspace. Every permanent database table has two extent sizes associated with it. The initial-extent size is the

number of kilobytes allocated to the table when it is first created. The next-extent size is the number of kilobytes allocated to the table when the initial extent, and every extent thereafter, becomes full.

Blobspaces do not use extents.

For specific instructions on how to specify and calculate the size of an extent, see your *IBM Informix Performance Guide*.

Extent size:

The default size for first and next extents is 16 kilobytes. If this transforms to fewer than 4 pages in a particular dbspace, the database server uses the minimum extent size of 4 pages. If a dbspace has a size of 8 kilobytes, which transforms to 2 pages, the database server increases the extent size to 32 kilobytes.

The maximum size of an extent is 2^{31} pages, equivalent to the maximum chunk size.

If the chunk is smaller than the maximum size, the maximum extent size depends on the contiguous space available in the chunk.

Tblspaces that hold *index fragments* follow different rules for extent size. The database server bases the extent size for these tblspaces on the extent size for the corresponding table fragment. The database server uses the ratio of the row size to index key size to assign an appropriate extent size for the index tblspace (see the sections on estimating index page size and fragmenting table indexes in the *IBM Informix Performance Guide*).

The maximum number of extents for a partition is 32767.

Page Types Within a Table Extent:

Within the extent, individual pages contain different types of data. Extent pages for a table can be separated into the following categories:

- Data pages
Data pages contain the data rows for the table.
- Bitmap pages
Bitmap pages contain control information that monitors the fullness of every page in the extent.
- Blobpages
Blobpages contain TEXT and BYTE data that is stored with the data rows in the dbspace. TEXT and BYTE data that resides in a blobspace is stored in blobpages, a structure that is completely different than the structure of a dbspace blobpage.
- Free pages
Free pages are pages in the extent that are allocated for tblspace use, but whose function has not yet been defined. Free pages can be used to store any kind of information: data, including TEXT or BYTE data types; index; or bitmap.

Figure 4-8 on page 4-9 illustrates the possible structure of a nonfragmented table with an initial-extent size of 8 pages and a next-extent size of 16 pages.

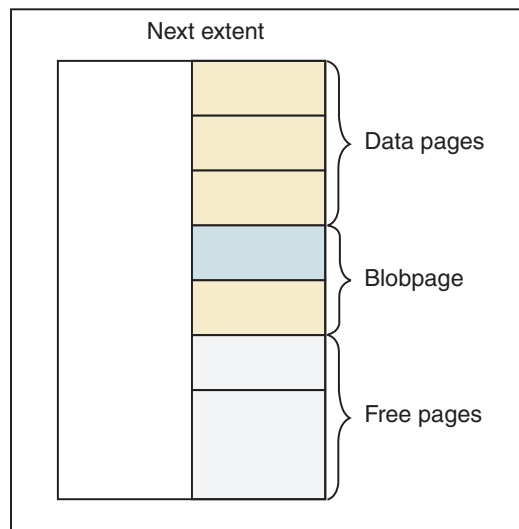
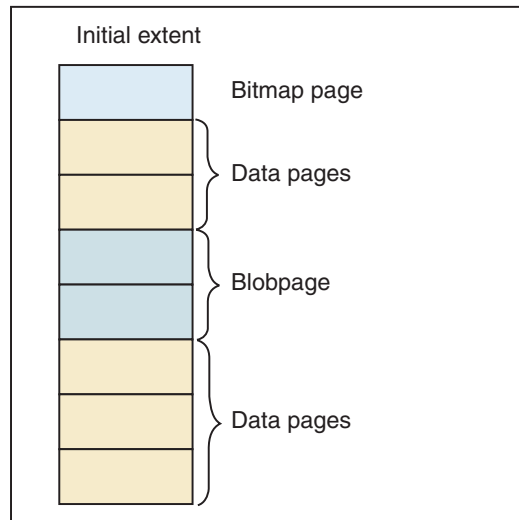


Figure 4-8. Extent Structure of a Table

Page Types Within an Index Extent:

The database server stores index pages into different tablespaces than the table with which it is associated. Within the extent, individual index pages contain different types of data. Index pages can be separated into the following categories:

- Index pages (root, branch, and leaf pages)
Index pages contain the index information for the table.
- Bitmap pages
Bitmap pages contain control information that monitors the fullness of every page in the extent.
- Free pages
Free pages are pages in the extent that are allocated for tablespace use, but whose function has not yet been defined. Free pages can be used to store any kind of information: data, index, TEXT or BYTE data, or bitmap.

All indexes are detached unless you explicitly specify attached indexes.

Important: An extent that is allocated for a table fragment does not contain index pages. Index pages for a fragmented table always reside in a separate tblspace. For more information, see fragmenting table indexes in the chapter on table fragmentation and PDQ in the *IBM Informix Administrator's Guide*.

Figure 4-9 illustrates the extent structure of an index.

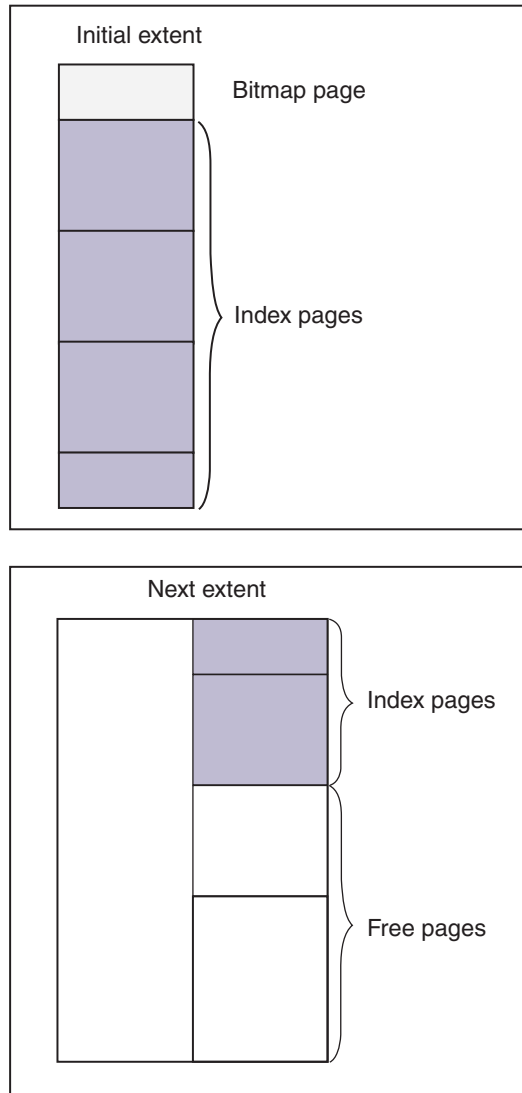


Figure 4-9. Extent Structure of an Index

Next-Extent Allocation

After the initial extent fills, the database server attempts to allocate another extent of contiguous disk space. The procedure that the database server follows is referred to as next-extent allocation.

Extents for a tblspace are tracked as one component of the tblspace **tblspace** information for the table. The maximum number of extents allocated for any tblspace is application and machine dependent because it varies with the amount of space available on the tblspace **tblspace** entry.

Next-Extent Size:

The number of kilobytes that the database server allocates for a next extent is, in general, equal to the size of a next extent, as specified in the SQL statement `CREATE TABLE`. However, the actual size of the next-extent allocation might deviate from the specified size because the allocation procedure takes into account the following three factors:

- Number of existing extents for this `tblspace`
- Availability of contiguous space in the chunk and `dbspace`
- Location of existing `tblspace` extents

The effect of each of these factors on next-extent allocation is explained in the paragraphs that follow and in Figure 4-10 on page 4-12.

Extent size doubling:

For permanent tables or user-defined temporary tables, the size of the next extent for every allocation is automatically doubled. The size doubles up to 128 kilobytes (KB). For example, if you create a table with the `NEXT SIZE` equal to 15 KB, the database server allocates the first extent at a size of 15 KB. The next extent is allocated at 30 KB, and the extent after that is allocated at 60 KB. When the extent size reaches 128 KB, the size is doubled only when the remaining space in the table is less than 10% of the total allocated space in the table.

For system-created temporary tables, the next-extent size begins to double after 4 extents have been added.

Lack of Contiguous Space:

If the database server cannot find available contiguous space in the first chunk equal to the size specified for the next extent, it extends the search to the next chunk in the `dbspace`. Extents are not allowed to span chunks.

If the database server cannot find adequate contiguous space anywhere in the `dbspace`, it allocates to the table the largest available amount of contiguous space. (The minimum allocation is four pages. The default value is eight pages.) No error message is returned if an allocation is possible, even when the amount of space allocated is less than the requested amount.

Merge of Extents for the Same Table:

If the disk space allocated for a next extent is physically contiguous with disk space already allocated to the same table, the database server allocates the disk space but does not consider the new allocation as a separate extent. Instead, the database server extends the size of the existing contiguous extent. Thereafter, all disk-space reports reflect the allocation as an extension of the existing extent. That is, the number of extents reported is always the number of physically distinct extents, not the number of times a next extent has been allocated plus one (the initial extent). Figure 4-10 on page 4-12 illustrates extent-allocation strategies.

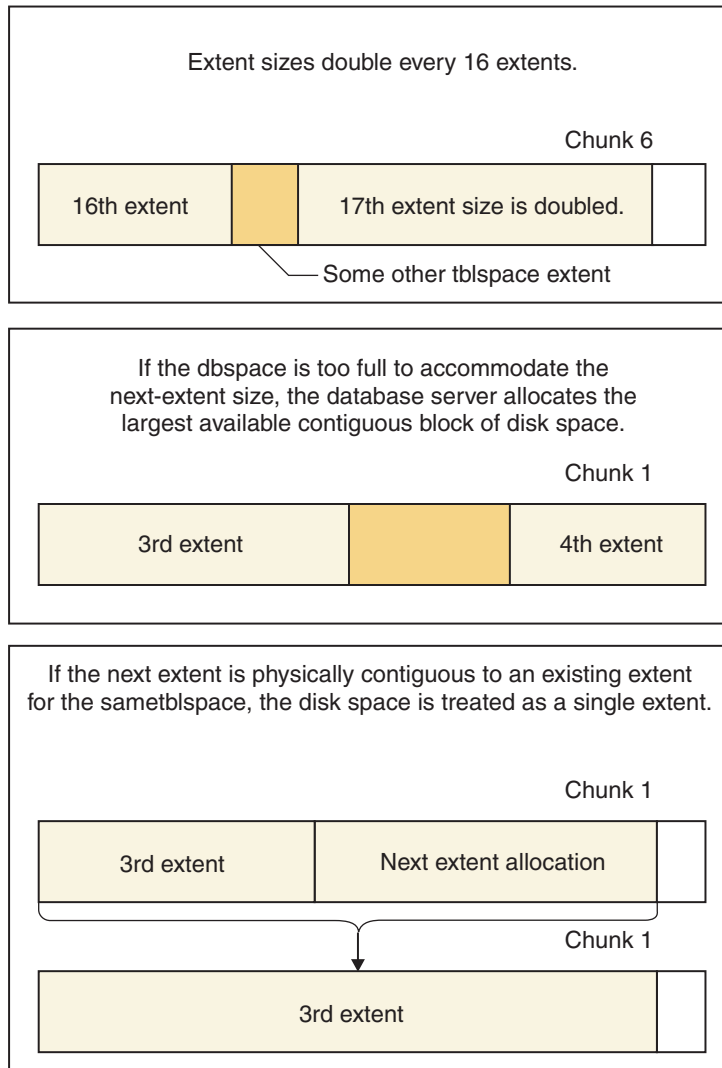


Figure 4-10. Next-Extent Allocation Strategies

After disk space is allocated to a tblspace as part of an extent, the space remains dedicated to that tblspace even if the data contained in it is deleted. For alternative methods of reclaiming this empty disk space, see your *IBM Informix Performance Guide*.

Structure and Storage of a Dbspace Page

The basic unit of database server I/O is a page. Page size might vary among computers.

In Informix, the page size depends on the operating system.

Rows in Nonfragmented Tables

The database server can store rows that are longer than a page. The database server also supports the VARCHAR data type, which results in rows of varying length. As a result, rows do not conform to a single format.

Rows within a table are not necessarily the same length if the table contains one or more columns of type VARCHAR. In addition, the length of a row in such a table might change when an end user modifies data contained in the VARCHAR column.

The length of a row can be greater than a page.

TEXT and BYTE data is not stored within the data row. Instead, the data row contains a 56-byte descriptor that points to the location of the data. The descriptor can point to a dbspace page.

The descriptor can point to a blobpage blobpage. If you are using the Optical Subsystem, the descriptor can also point to an optical-storage subsystem.

For instructions about how to estimate the length of fixed-length and variable-length data rows, see your *IBM Informix Performance Guide*.

Definition of Rowid: Informix uses two different types of rowids to identify data in tables:

- *Serial rowid*

These rowids are fields in a table and are assigned to tables created with the WITH ROWID option.

- *Internal rowid*

The database server identifies each data row in a table with a unique internal rowid. This rowid identifies the location of the row within the dbspace.

To obtain the internal rowids for a table, use the **oncheck -pD** option. For more information, see “**oncheck -cd** and **oncheck -cD** commands: Check pages” on page 8-8.

In a nonfragmented table, the term *rowid* refers to a unique 4-byte integer that defines the physical location of the row in the table. The page that contains the first byte of the data row is the page that is specified by the rowid. This page is called the data row *home page*.

Fragmented tables can also have rowids, but they are implemented in a different way. For more information on this topic, see “Rows in Fragmented Tables.”

Use of Rowids: Every data row in a nonfragmented table is uniquely identified by an unchanging rowid. When you create an index for a nonfragmented table, the rowid is stored in the index pages associated with the table to which the data row belongs. When the database server requires a data row, it searches the index to find the key value and uses the corresponding rowid to locate the requested row. If the table is not indexed, the database server might sequentially read all the rows in the table.

Eventually, a row might outgrow its original storage location. If this occurs, a *forward pointer* to the new location of the data row is left at the position defined by the rowid. The forward pointer is itself a rowid that defines the page and the location on the page where the data row is now stored.

Rows in Fragmented Tables

Unlike rows in a nonfragmented table, the database server does *not* assign a rowid to rows in fragmented tables. If you want to access data by rowid, you must explicitly create a rowid column as described in your *IBM Informix Performance*

Guide. If user applications attempt to reference a rowid in a fragmented table that does not contain a rowid that you explicitly created, the database server returns an appropriate error code to the application.

Access to Data in Fragmented Tables with Rowid: From the viewpoint of an application, the functionality of a rowid column in a fragmented table is identical to the rowid of a nonfragmented table. However, unlike the rowid of a nonfragmented table, the database server uses an index to map the rowid to a physical location.

When the database server accesses a row in a fragmented table using the rowid column, it uses this index to look up the physical address of the row before it attempts to access the row. For a nonfragmented table, the database server uses direct physical access without an index lookup. As a consequence, accessing a row in a fragmented table using rowid takes slightly longer than accessing a row using rowid in a nonfragmented table. You should also expect a small performance impact on the processing of inserts and deletes due to the cost of maintaining the rowid index for fragmented tables.

Primary-key access can lead to significantly improved performance in many situations, particularly when access is in parallel.

Recommendations on Use of Rowid

It is recommended that application developers use primary keys as a method of access rather than rowids. Because primary keys are defined in the ANSI specification of SQL, using them to access data makes your applications more portable.

For a complete description on how to define and use primary keys to access data, see the *IBM Informix Guide to SQL: Reference* and the *IBM Informix Guide to SQL: Tutorial*.

Data-Row Format and Storage

The variable length of a data row has the following consequences for row storage:

- A page might contain one or more whole rows.
- A page might contain portions of one or more rows.
- A page might contain a combination of whole rows and partial rows.
- An updated row might increase in size and become too long to return to its original storage location in a row.

The following paragraphs describe the guidelines that the database server follows during data storage.

Storage of Row:

To minimize retrieval time, rows are not broken across page boundaries unnecessarily. Rows that are shorter than a page are always stored as whole rows. A page is considered *full* when the count of free bytes is less than the number of bytes needed to store a row of maximum size.

Location of Rows:

When the database server receives a row that is longer than a page, the row is stored in as many whole pages as required. The database server then stores the trailing portion in less than a full page.

The page that contains the first byte of the row is the row home page. The number of the home page becomes the logical page number contained in the rowid. Each full page that follows the home page is referred to as a big-remainder page. If the trailing portion of the row is less than a full page, it is stored on a remainder page.

After the database server creates a remainder page to accommodate a long row, it can use the remaining space in this page to store other rows.

Figure 4-11 illustrates the concepts of home page, big-remainder page, and remainder page.

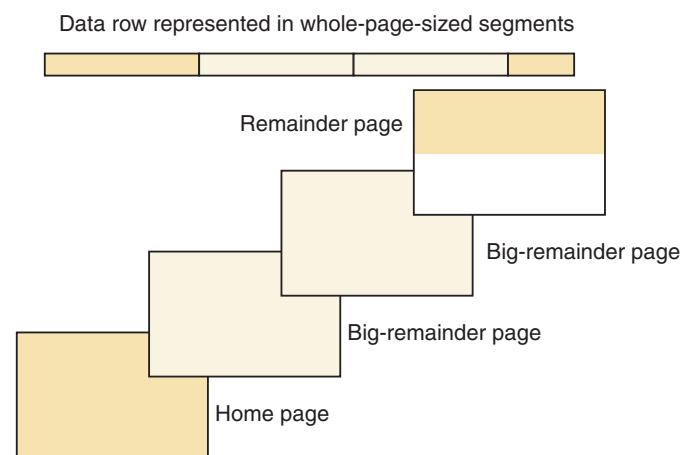


Figure 4-11. Remainder Pages

Page Compression:

Over time, the free space on a page can become fragmented. When the database server attempts to store data, it first checks row length against the number of free bytes on a page to determine if the row fits. If adequate space is available, the database server checks if the page contains adequate contiguous free space to hold the row (or row portion). If the free space is not contiguous, the database server calls for page compression.

Structure of Fragmented Tables

Although table fragmentation is transparent to applications, as database server administrator you should be aware of how the database server allocates disk space for table fragments and how the database server identifies rows in those fragments.

Each table fragment has its own *tblspace* with a unique *tblspace_id* or *fragment_id*. Figure 4-12 on page 4-16 shows the disk allocation for a fragmented table that resides in named fragments of the same *dbspace*.

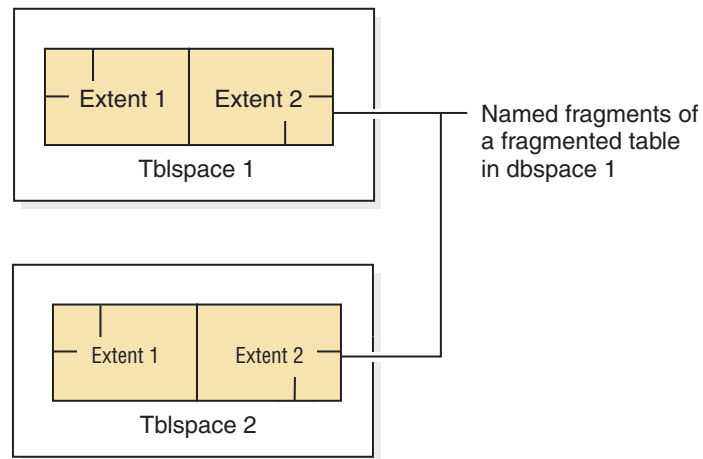


Figure 4-12. Disk Structures for a Fragmented Table

Attached Indexes

With an attached index, the index and data are fragmented in the same way. You can decide whether to store the index pages with the corresponding data pages in the same dbspace or store them in separate dbspaces. For information on choosing a fragmentation strategy, see the *IBM Informix Performance Guide*.

Detached Indexes

For detached indexes, the table fragment and index fragment are stored in tablespaces in separate dbspaces.

Structure of B-Tree Index Pages

This section provides general information about the structure of B-tree index pages. It is designed as an overview for the interested reader. For more information on B-tree indexes, see your *IBM Informix Performance Guide*.

Definition of B-tree terms

The database server uses a B-tree structure to organize index information.

Figure 4-13 on page 4-17 shows that a fully developed B-tree index is composed of the following three different types of index pages or nodes:

- One *root node*
A root node contains node pointers to branch nodes.
- Two or more *branch nodes*
A branch node contains pointers to leaf nodes or other branch nodes.
- Many *leaf nodes*
A leaf node contains index items and horizontal pointers to other leaf nodes.

Each node serves a different function. The following sections describe each node and the role that it plays in indexing.

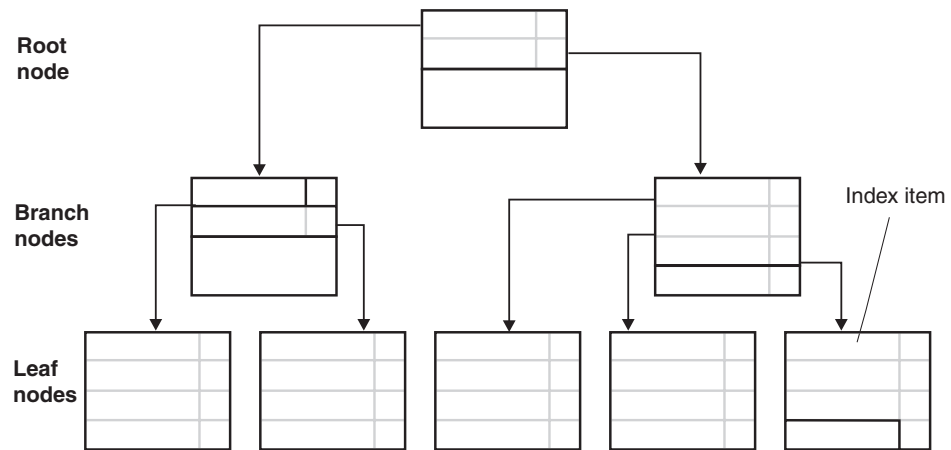


Figure 4-13. Full B-Tree Structure

Index items

The fundamental unit of an index is the *index item*. An index item contains a key value that represents the value of the indexed column for a particular row. An index item also contains rowid information that the database server uses to locate the row in a data page.

Index nodes

A node is an index page that stores a group of index items.

Forest of trees indexes as alternatives to traditional B-Tree indexes

Unlike a traditional B-tree index, a forest of trees index is a large B-tree index that is divided into smaller subtrees with multiple root nodes and fewer levels. You can create a forest of trees index as an alternative to a B-tree index when you want to alleviate root node contention and allow more concurrent users to access the index without waiting.

Related concepts:

[Forest of trees indexes \(Performance Guide\)](#)

Logical Storage of Indexes

This section presents an overview of how the database server creates and fills an index.

Creation of Root and Leaf Nodes: When you create an index for an empty table, the database server allocates a single index page. This page represents the root node and remains empty until you insert data in the table.

At first, the root node functions in the same way as a leaf node. For each row that you insert into the table, the database server creates and inserts an index item in the root node. Figure 4-14 on page 4-18 illustrates how a root node appears before it fills.

Root node 1	
Albertson	rowid information
Baxter	rowid information
Beatty	rowid information
Currie	rowid information
Keyes	rowid information
Lawson	rowid information
Mueller	rowid information

Figure 4-14. Root Node

When the root node becomes full of index items, the database server splits the root node by performing the following steps:

- Creates two leaf nodes
- Moves approximately half of the root-node entries to each of the newly created leaf nodes
- Puts pointers to leaf nodes in the root node

As you add new rows to a table, the database server adds index items to the leaf nodes. When a leaf node fills, the database server creates a new leaf node, moves part of the contents of the full index node to the new node, and adds a node pointer to the new leaf node in the root node.

For example, suppose that leaf node 3 in Figure 4-15 becomes full. When this situation occurs, the database server adds yet another leaf node. The database server moves part of the records from leaf node 3 to the new leaf node, as Figure 4-15 shows.

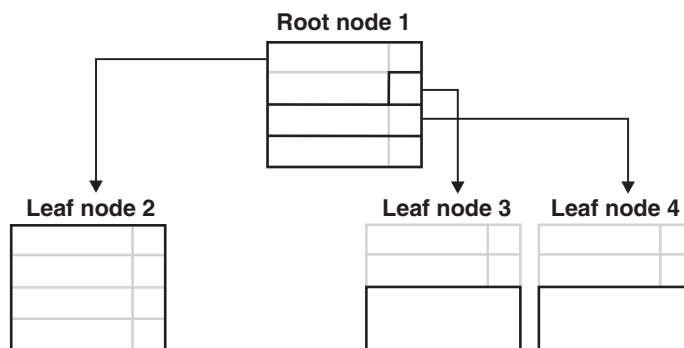


Figure 4-15. Leaf Node 4 Created After Leaf Node 3 Fills

Creation of branch nodes: Eventually, as you add rows to the table, the database server fills the root node with node pointers to all the existing leaf nodes. When the database server splits yet another leaf node, and the root node has no room for an additional node pointer, the following process occurs.

The database server splits the root node and divides its contents among two newly created branch nodes. As index items are added, more and more leaf nodes are split, causing the database server to add more branch nodes. Eventually, the root node fills with pointers to these branch nodes. When this situation occurs, the database server splits the root node again. The database server then creates yet another branch level between the root node and the lower branch level. This

process results in a four-level tree, with one root node, two branch levels, and one leaf level. The B-tree structure can continue to grow in this way to a maximum of 20 levels.

Branch nodes can point either to other branch nodes below them (for large indexes of four levels or more) or to leaf nodes. In Figure 4-16, the branch node points to leaf nodes only. The first item in the left branch node contains the same key value as the largest item in the leftmost leaf node and a node pointer to it. The second item contains the largest item in the next leaf node and a node pointer to it. The third item in the branch node contains only a pointer to the next higher leaf node. Depending on the index growth, this third item can contain the actual key value in addition to the pointer at a later point during the life span of the index.

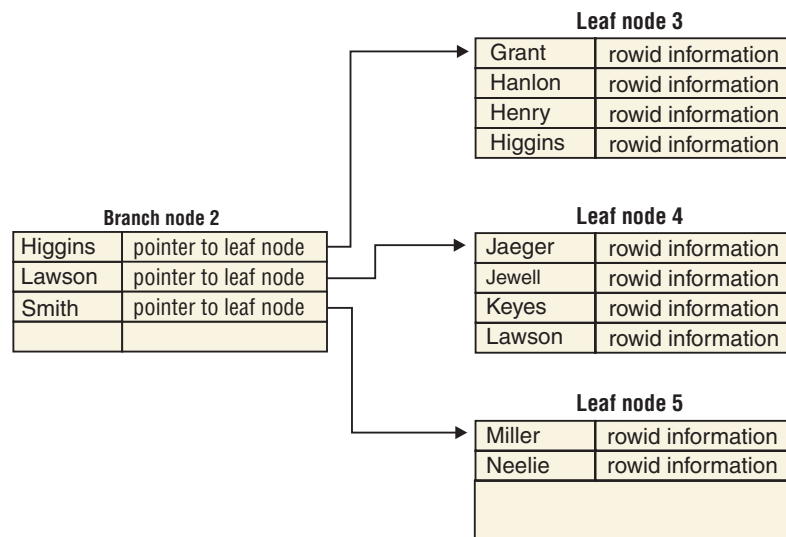


Figure 4-16. Typical Contents of a Branch Node

Duplicate Key Values: Duplicate key values occur when the value of an indexed column is identical for multiple rows. For example, suppose that the third and fourth leaf nodes of a B-tree structure contain the key value Smith. Suppose further that this value is duplicated six times, as Figure 4-17 illustrates.

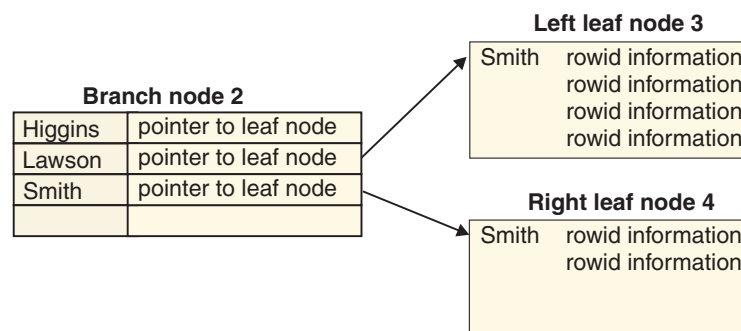


Figure 4-17. Leaf Nodes 3 and 4

The first item on the third leaf page contains the duplicate key value, Smith, and the rowid information for the first physical row in the table that contains the duplicate key value. To conserve space, the second item does not repeat the key

value Smith but instead contains just the rowid information. This process continues throughout the page; no other key values are on the leaf, only rowid information.

The first item on the fourth leaf page again contains the duplicated key value and rowid information. Subsequent items contain only rowid information.

Now consider the branch node. The third item in the branch node contains the same key value and rowid as the largest item in the third leaf node and a node pointer to it. The fourth item would contain only a node pointer to the fourth leaf node, thus saving the space of an additional duplicate key value.

Key-Value Locking: To increase concurrency, the database server supports *key-value* locking in the B-tree index. Key-value locking locks only the value of the key instead of the physical location in the B-tree index.

One of the most important uses for key-value locking is to assure that a unique key remains unique through the end of the transaction that deleted it. Without this protection mechanism, user A might delete a unique key within a transaction, and user B might insert a row with the same key before the transaction commits. This scenario makes rollback by user A impossible. Key-value locking prevents user B from inserting the row until the end of user A's transaction.

Adjacent Key Locking: With Repeatable Read isolation level, the database server is required to protect the *read set*. The read set consists of the rows that meet the filters in the WHERE clause of the query. To guarantee that the rows do not change, the database server obtains a lock on the index item that is adjacent to the right-most item of the read set.

Freed Index Pages: When the database server physically removes an index item from a node and frees an index page, the freed page is reused.

Filling Indexes: When you create an index, you can specify how densely or sparsely filled you want the index. The index fill factor is a percentage of each index page that will be filled during the index build. Use the FILLFACTOR option of the CREATE INDEX statement or the FILLFACTOR configuration parameter to set the fill factor. This option is particularly useful for indexes that you do not expect to grow after they are built. For additional information about the FILLFACTOR option of the CREATE INDEX statement, see the *IBM Informix Guide to SQL: Syntax*.

Calculating the Length of Index Items: For data types other than VARCHAR, the length of an index item is calculated by adding the length of the key value plus 5 bytes for each rowid information associated with the key value.

The key values in an index are typically of fixed length. If an index holds the value of one or more columns of the VARCHAR data type, the length of the key value is at least the sum of the length-plus-one of each VARCHAR value in the key.

In Informix, the maximum length of a key value is 390 bytes. The combined size of VARCHAR columns that make up a key must be less than 390, minus an additional byte for each VARCHAR column. For example, the key length of the index that the database server builds for the following statements equals 390, or ((255+1) + (133+1)):

```
CREATE TABLE T1 (c1 varchar(255, 10), c2 varchar(133, 10));  
CREATE INDEX I1 on T1(c1, c2);
```


Functional Indexes

A *functional index* is one in which all keys derive from the results of a function. If you have a column of pictures, for example, and a function to identify the predominant color, you can create an index on the result of the function. Such an index would enable you to quickly retrieve all pictures having the same predominant color, without re-executing the function.

A functional index uses the same B-tree structure as any other B-tree index. The only difference is that the determining function is applied during an insert or an update whenever the column that is the argument to the function changes. For more information on the nature of functional indexes, refer to your *IBM Informix Performance Guide*.

To create a functional index, use the CREATE FUNCTION and CREATE INDEX statements. For more information on these statements, refer to the *IBM Informix Guide to SQL: Syntax*.

Structure of R-Tree Index Pages

An index structure that relies on one-dimensional ordering of key values does not work for spatial data; for example, two dimensional geometric shapes such as circles, squares, and triangles. Efficient retrieval of spatial data, such as the data used in geographic information systems (GIS) and computer-aided design (CAD) applications, requires an access method that handles multidimensional data. The database server implements an R-tree index to access spatial data efficiently. For information about the structure of index pages, refer to the *IBM Informix R-Tree Index User's Guide*.

Storage of Simple Large Objects

This section explains the structures and storage techniques that the database server uses to store simple large objects (TEXT or BYTE data).

Structure of a Blobspace

When you create a blobspace, you can specify the effective size of the data pages, which are called blobpages. The blobpage size for the blobspace is specified when the blobspace is created. Blobpage size must be a multiple of page size. (For information on determining database server page size, see the chapter on managing disk space in the *IBM Informix Administrator's Guide*.) All blobpages within a blobspace are the same size, but the size of the blobpage can vary between blobspaces. Blobpage size can be greater than the page size because data stored in a blobspace is never written to the page-sized buffers in shared memory.

The advantage of customizing the blobpage size is storage efficiency. Within a blobspace, TEXT and BYTE data is stored in one or more blobpages, but simple large objects do not share blobpages. Storage is most efficient when the TEXT or BYTE data is equal to or slightly smaller than the blobpage size.

The blobspace free-map pages and bitmap pages are the size specified as a database server page, which enables them to be read into shared memory and to be logged.

When the blobspace is first created, it contains the following structures:

- Blobspace free-map pages
- The blobspace bitmap that tracks the free-map pages

- Unused blobpages

Structure of a Dbspace Blobpage

TEXT or BYTE data that is stored in the dbspace is stored in a blobpage. The structure of a dbspace blobpage is similar to the structure of a dbspace data page. The only difference is an extra 12 bytes that can be stored along with the TEXT or BYTE data in the data area.

Simple large objects can share dbspace blobpages if more than one simple large object can fit on a single page, or if more than one trailing portion of a simple large object can fit on a single page.

For a discussion of how to estimate the number of dbspace blobpages needed for a specific table, see your *IBM Informix Performance Guide*.

Each segment of TEXT or BYTE data stored in a dbspace page might be preceded by up to 12 bytes of information that does not appear on any other dbspace page. These extra bytes are overhead.

Simple-Large-Object Storage and the Descriptor

Data rows that include TEXT or BYTE data do not include the data in the row itself. Instead, the data row contains a 56-byte descriptor with a forward pointer (rowid) to the location where the first segment of data is stored.

The descriptor can point to one of the following items:

- A page (if the data is stored in a dbspace)
- A blobpage (if the data is stored in a blobpage)
- An optical platter (if you are using the Optical Subsystem)

Creation of Simple Large Objects

When a row that contains TEXT or BYTE data is to be inserted, the simple large objects are created first. After the simple large objects are written to disk (or optical medium), the row is updated with the descriptor and inserted.

Deletion or Insertion of Simple Large Objects

The database server cannot modify simple large objects. It can only insert or delete them. Deleting a simple large object means that the database server frees the space consumed by the deleted object for reuse.

When TEXT or BYTE data is updated, a new simple large object is created, and the data row is updated with the new blob descriptor. The old image of the row contains the descriptor that points to the obsolete value for the simple large object. The space consumed by the obsolete simple large object is freed for reuse after the update is committed. Simple large objects are automatically deleted if the rows that contain their blob descriptors are deleted. (Blobpages that stored a deleted simple large object are not available for reuse until the logical log that contains the original INSERT record for the deleted simple large object is backed up. For more information, see backing up logical-log files to free blobpages in the chapter on what is the logical log in the *IBM Informix Administrator's Guide*.)

Size Limits for Simple Large Objects

The largest simple large object that the blob descriptor can accommodate is $(2^{31} - 1)$, or about 2 gigabytes.

Blobspace Page Types

Every blobspace chunk contains three types of pages:

- A blobspace free-map page
- A bitmap page
- Blobpages

Blobspace Free-Map Page

The blobspace free-map page identifies unused blobpages so that the database server can allocate them as part of simple-large-object creation. When a blobpage is allocated, the free-map entry for that page is updated. All entries for a single simple large object are linked.

A blobspace free-map page is the size of one database server page. Each entry on a free-map page is 8 bytes, stored as two 32-bit words, as follows:

- The first bit in the first word specifies whether the blobpage is free or used.
- The next 31 bits in the first word identify the logical-log file that was current when this blobpage was written. (This information is needed for logging TEXT or BYTE data.)
- The second word contains the tblspace number associated with the simple large object stored on this page.

The number of entries that can fit on a free-map page depends on the page size of your computer. The number of free-map pages in a blobspace chunk depends on the number of blobpages in the chunk.

Blobspace Bitmap Page

The blobspace bitmap page tracks the fullness and number of blobspace free-map pages in the chunk. Each blobspace bitmap page is capable of tracking a quantity of free-map pages. The size of the blobspace bitmap page depends on the size of the system page. If the system page is 2K, the blobspace bitmap page can track 2,032,128 blobpages. If the system page is 4K, the blobspace bitmap page can track 8,258,048 blobpages.

Blobpage

The blobpage contains the TEXT or BYTE data. Blobpage size is specified by the database server administrator who creates the blobspace. Blobpage size is specified as a multiple of the page size.

Structure of a Blobspace Blobpage

The storage strategy used to store simple large objects in a blobspace differs from the dbspace storage strategy. The database server does not combine whole simple large objects or portions of a simple large object on a single blobspace blobpage.

For example, if blobpage blobpages are 24 kilobytes each, a simple large object that is 26 kilobytes is stored on two 24-kilobyte pages. The extra 22 kilobytes of space remains unused.

The structure of a blobpage includes a blobpage header, the TEXT or BYTE data, and a page-ending time stamp. The blobpage header includes, among other information, the page-header time stamp and the blob time stamp associated with the forward pointer in the data row. If a simple large object is stored on more than one blobpage, a forward pointer to the next blobpage and another blob time stamp are also included in the blobpage header.

Sbospace Structure

An sbospace is similar to a blobspace except that it holds smart large objects.

When an sbospace is created in a database, it contains an sbospace descriptor. Each sbospace chunk contains the following structures:

- Sbspace chunk descriptors
- Chunk free-page list
- An sbospace metadata area (up to one for each chunk)
- Reserved data areas (up to two for each chunk)
- User-data areas (up to two for each chunk)

For best performance, it is recommended that the metadata area be located in the middle of the sbospace. The database server automatically places the metadata area in the correct location. However, to specify the location of the metadata area, specify the **-Mo** flag in the **onspaces** command.

If you do not specify the size of the metadata area in the **-Ms** flag of the **onspaces** command, the database server uses the value of `AVG_LO_SIZE` (defaults to 8 kilobytes) to calculate the size of the metadata area. For more information, see “Creating an Sbospace with the **-Df** option” on page 19-12.

Normally, you can let the system calculate the metadata size for you. If you want to estimate the size of the metadata area, see the chapter on table performance considerations in the *IBM Informix Performance Guide*.

Figure 4-18 on page 4-25 illustrates the chunk structure of an sbospace as it appears immediately after the sbospace is created. Each reserved area can be allocated to either the user-data or metadata area. Reserved areas are always within the user-data area of the chunk.

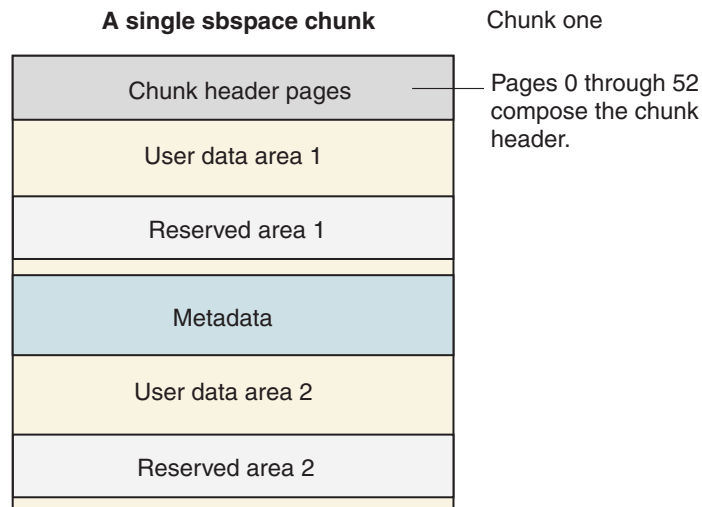


Figure 4-18. A Single Sbospace Chunk

Because the chunk in Figure 4-18 is the first in the sbospace, it contains an sbospace descriptor. The chunk descriptor tbspace in **chunk one** contains information about chunk one and all chunks added to the sbospace thereafter.

Structure of the metadata area

An sbospace contains a metadata area for each chunk in the sbospace.

As with the chunk header pages, four areas are exclusive to the first chunk in a sbospace: the sbospace descriptor tbspace, the chunk adjunct tbspace, and the level-1 and level-2 archive tbspaces. The tbspace header section contains a tbspace header for each of these tbspaces (notably excluding the tbspace **tbspace**). Figure 4-19 shows the layout of the metadata in the single-chunk sbospace.

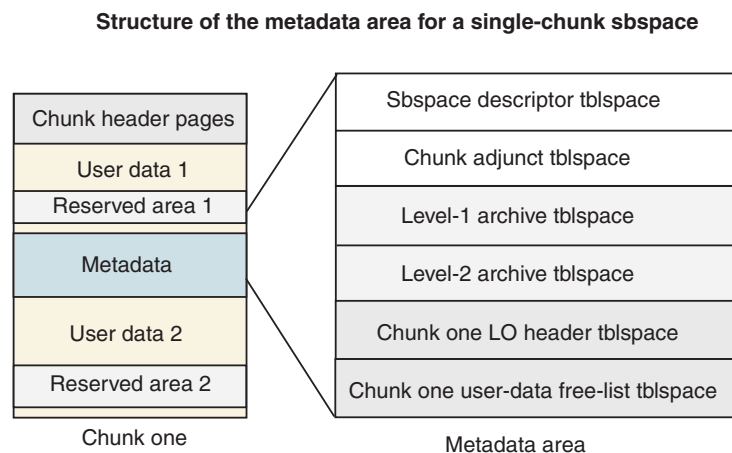


Figure 4-19. Structure of the metadata area for a single-chunk sbospace

When you specify the sbospace name in the **oncheck -ps** option, you can display the number of pages allocated and used for each tbspace in the metadata area.

The following items describe how the metadata area grows:

- The sbospace descriptor tbspace does not grow.

- The chunk adjunct tblspace grows as chunks are added.
- The LO header tblspace grows as large objects are added to the chunk.
- The tblspace for user-data free list grows if free spaces in the chunk are heavily fragmented.

Sbpage Structure

Each sbpage is composed of three elements: an sbpage header, the actual user data itself, and an sbpage trailer. Figure 4-20 shows the structure of an sbpage. The sbpage header consists of the standard page header. The sbpage trailer is used to detect an incomplete write on the page and to detect page corruption.

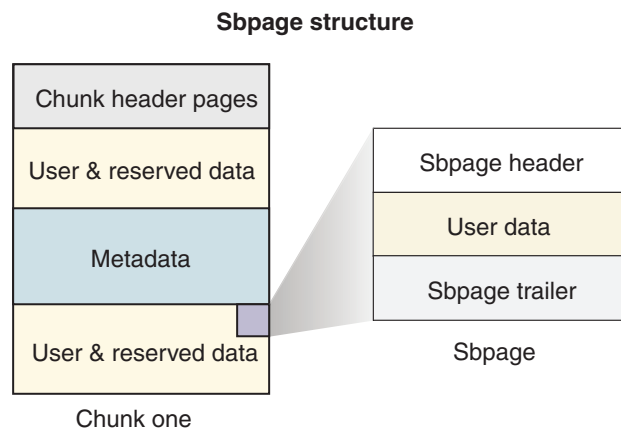


Figure 4-20. Sbpage Structure

Multiple Chunk Sbspace

Figure 4-21 illustrates a possible configuration for a three-chunk sbspace. In this example, **chunk two** contains no metadata of its own. Metadata information for **chunk two** is stored in the metadata area of **chunk one**.

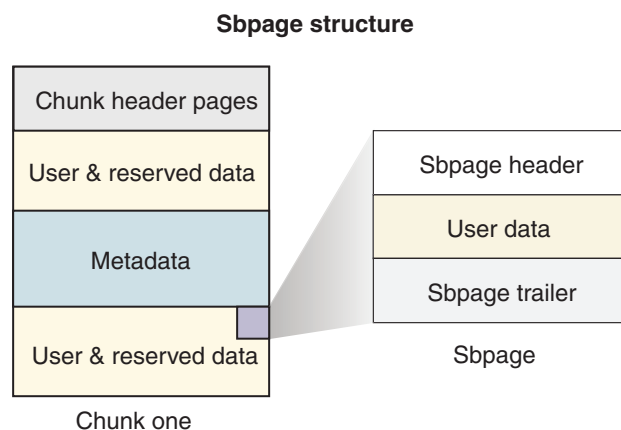


Figure 4-21. Multiple-Chunk Sbspace Structure

The user-data area in **chunk one** of the example is actually optional. **Chunk one** could contain metadata for all other chunks in the sbspace.

Time Stamps

The database server uses a time stamp to identify a time when an event occurred relative to other events of the same kind. The time stamp is not a literal time that refers to a specific hour, minute, or second. It is a 4-byte integer that the database server assigns sequentially.

Database and Table Creation: What Happens on Disk

This section explains how the database server stores data related to the creation of a database or table and allocates the disk structures that are necessary to store your data.

Database Creation

After the root dbspace exists, users can create a database. The paragraphs that follow describe the major events that occur on disk when the database server adds a new database.

Disk-Space Allocation for System Catalog Tables

The database server searches the chunk free-list pages in the dbspace, looking for free space in which to create the system catalog tables. For each system catalog table, in turn, the database server allocates eight contiguous pages, the size of the initial extent of each system catalog table. The tables are created individually and do not necessarily reside next to each other in the dbspace. They can be located in different chunks. As adequate space is found for the initial extent of each table, the pages are allocated, and the associated chunk free-list page is updated.

Tracking of System Catalog Tables

The database server tracks newly created databases in the database tblspace, which resides in the root dbspace. An entry describing the database is added to the database tblspace in the root dbspace. (See “Structure of the Database Tblspace” on page 4-7.) For each system catalog table, the database server adds a one-page entry to the tblspace **tblspace** in the dbspace where the database was built. (See “Structure of the Tblspace Tblspace” on page 4-4.) Figure 4-22 illustrates the relationship between the database tblspace entry and the location of the **systables** system catalog table for the database.

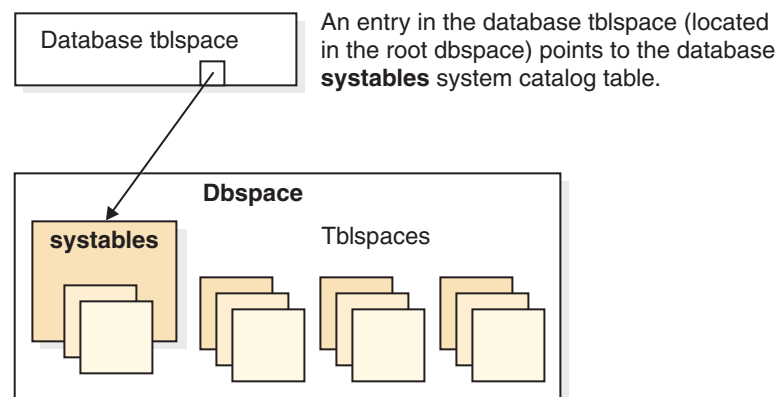


Figure 4-22. New Databases

For instructions on how to list your databases after you create them, see monitoring databases in the chapter on managing database-logging status in the *IBM Informix Administrator's Guide*.

Table Creation

After the root dbspace exists, and a database has been created, users with the necessary SQL privileges can create a database table. When users create a table, the database server allocates disk space for the table in units called extents (see what is an extent in the chapter on where data is stored in the *IBM Informix Administrator's Guide*). The paragraphs that follow describe the major events that occur when the database server creates a table and allocates the initial extent of disk space.

Disk-Space Allocation

The database server searches the chunk free-list pages in the dbspace for contiguous free space equal to the initial extent size for the table. When adequate space is found, the pages are allocated, and the associated chunk free-list page is updated.

If the database server cannot find adequate contiguous space anywhere in the dbspace, it allocates to the table the largest available amount of contiguous space. No error message is returned if an allocation is possible, even when the amount of space allocated is less than the requested amount. If the minimum extent size cannot be allocated, an error is returned. (Extents cannot span two chunks.)

Entry in the Tblspace Tblspace

The database server adds a one-page entry for this table to the tblspace **tblspace** in this dbspace. The tblspace number assigned to this table is derived from the logical page number in the tblspace **tblspace** where the table is described. See “Tblspace Numbers” on page 4-5.

The tblspace number indicates the dbspace where the tblspace is located. Tblspace extents can be located in any of the dbspace chunks.

If you must know exactly where the tblspace extents are located, execute the **oncheck -pe** command for a listing of the dbspace layout by chunk.

Entries in the System Catalog Tables

The table itself is fully described in entries stored in the system catalog tables for the database. Each table is assigned a table identification number or *tabid*. The *tabid* value of the first user-defined table object in a database is always 100. (The object whose *tabid* = 100 might also be a view, synonym, or a sequence.) For a complete discussion of the system catalog, see the *IBM Informix Guide to SQL: Reference*.

A table can be located in a dbspace that is different than the dbspace that contains the database. The tblspace itself is the sum of allocated extents, not a single, contiguous allocation of space. The database server tracks tblspaces independently of the database.

Creation of a Temporary Table

The tasks involved in creating temporary tables are similar to the tasks that the database server performs when it adds a new permanent table. The key difference is that temporary tables do not receive an entry in the system catalog for the database. For more information, see the section defining a temporary table, in the chapter on where data is stored in the *IBM Informix Administrator's Guide*.

Chapter 5. Interpreting Logical-Log Records

In This Chapter

To display the logical-log records that the logical-log files contain, use the **onlog** utility.

This chapter provides the following information:

- Brief guidance on reading logical-log records
- A listing of the different logical-log record types

In general, you do not need to read and interpret your logical-log files. However, **onlog** output is useful in debugging situations. For example, you might want to use **onlog** to track a specific transaction or to see what changes the database server made to a specific tblspace. You can also use **onlog** to investigate the cause of an error that occurs during a rollforward. For more information, see “onlog: Display Logical-Log Contents” on page 13-1.

About Logical-Log Records

Most SQL statements generate multiple logical-log records. Interpreting logical-log records is more complicated when the database server records the following events in the logical log:

- A transaction that drops a table or index
- A transaction that rolls back
- A checkpoint in which transactions are still active
- A distributed transaction

The following sections discuss the logical-log records for these events.

Transactions That Drop a Table or Index

Once the database server drops a table or index from a database, it cannot roll back that drop operation. If a transaction contains a DROP TABLE or DROP INDEX statement, the database server handles this transaction as follows:

1. The database server completes all the other parts of the transaction and writes the relevant logical-log records.
2. The database server writes a BEGCOM record to the logical log and the records associated with the DROP TABLE or DROP INDEX (DINDEX, for example).
3. The database server writes a COMMIT record.

If the transaction is terminated unexpectedly after the database server writes the BEGCOM record to the logical log, the database server rolls *forward* this transaction during recovery because it cannot roll back the drop operation.

Transactions That Are Rolled Back

When a rollback occurs, the database server generates a compensation-log record (CLR) for each record in the logical log that is rolled back. The database server uses the CLRs if a system failure takes place *during a rollback*. The CLRs provide

the database server with information on how far the rollback progressed before the failure occurred. In other words, the database server uses the CLRs to log the rollback.

If a CLR contains the phrase includes next record, the next log record that is printed is included within the CLR log record as the compensating operation. Otherwise, you must assume that the compensating operation is the logical undo of the log record to which the **link** field of the CLR points.

Checkpoints with Active Transactions

If any transactions are active at the time of a checkpoint, checkpoint records include subentries that describe each of the active transactions using the following columns:

- Log begin (decimal format)
- Transaction ID (decimal format)
- Unique log number (decimal format)
- Log position (hexadecimal format)
- User name

Distributed Transactions

When distributed transactions (transactions that span multiple database servers) generate log records, they are slightly different than nondistributed transactions. You might need to read and interpret them to determine the state of the transaction on both database servers if a failure occurs as a transaction was committing.

The following log records are involved in distributed transactions:

- BEGPREP
- ENDTRANS
- HEURTX
- PREPARE
- TABLOCKS

For more information about this type of logical-log record, see the material on two-phase commit and logical-log records in the *IBM Informix Administrator's Guide*.

If you are performing distributed transactions with TP/XA, the database server uses an XAPREPARE record instead of a PREPARE record.

Logical-Log Record Structure

Each logical-log record has *header* information. Depending on the record type, additional columns of information also appear in the output, as explained in “Logical-Log Record Types and Additional Columns” on page 5-3.

Logical-Log Record Header

Table 5-1 on page 5-3 contains sample output to illustrate the header columns that display for a logical-log record.

Table 5-1. Sample Output from onlog

addr	len	type	xid	id	link
2c018	32	BEGIN	6	3	0
2c038	140	HDELETE	6	0	2c018
2c0c4	64	DELITEM	6	0	2c038
2c104	40	DELITEM	6	0	2c0c4
2c12c	72	HDELETE	6	0	2c104
2c174	44	DELITEM	6	0	2c12c
2c1a0	72	HDELETE	6	0	2c174
2c1e8	44	DELITEM	6	0	2c1a0
2c214	64	HDELETE	6	0	2c1e8
2c254	56	DELITEM	6	0	2c214
2c28c	48	DELITEM	6	0	2c254
2c2bc	24	PERASE	6	0	2c28c
2c2d4	20	BEGCOM	6	0	2c2bc
2c2e8	24	ERASE	6	0	2c2d4
2c300	28	CHFREE	6	0	2c2e8
2c31c	24	COMMIT	6	0	2c300

Table 5-2 defines the contents of each header column.

Table 5-2. Definition of onlog Header Columns

Header Field	Contents	Format
addr	Log-record address (log position)	Hexadecimal
len	Record length in bytes	Decimal
type	Record-type name	ASCII
xid	Transaction number	Decimal
id	Logical-log number	Decimal
link	Link to the previous record in the transaction	Hexadecimal

Logical-Log Record Types and Additional Columns

In addition to the six header columns that display for every record, some record types display additional columns of information. The information that appears varies, depending on record type. The following table lists all the record types and their additional columns.

The **Action** column indicates the type of database server action that generated the log entry. The **Additional Columns** and **Format** columns describe what information appears for each record type in addition to the header described in “Logical-Log Record Header” on page 5-2.

Table 5-3. Logical-Log Record Types

Record Type	Action	Additional Columns	Format
ADDCHK	Add chunk.	chunk number	Decimal
		chunk name	ASCII
ADDDBS	Add dbspace.	dbspace name	ASCII
ADDITEM	Add item to index.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		logical page	Decimal
		key number	Decimal
		key length	Decimal
ADDLOG	Add log.	log number	Decimal
		log size (pages)	Decimal
		pageno	Hexadecimal
ALLOCGENPG	Allocate a generic page.	tblspace ID	Decimal
		rowid	Decimal
		slot flags and length	Decimal
		page version if delete	Decimal
		flags, vimage record	Decimal
		rowid for previous	Decimal
		data	ASCII
ALTERDONE	Alter of fragment complete.	tblspace ID	Hexadecimal
		physical page number previous page	Hexadecimal
		logical page number	Decimal
		version of alter	Decimal
ALTSPCOLSNEW	Changed columns in an alter table.	number of columns	Decimal
		special column list	array
ALTSPCOLSOLD	Changed columns in an alter table.	number of columns	Decimal
		special column list	array
BADIDX	Bad index	tblspace ID	Hexadecimal
BEGCOM	Begin commit.	(None)	(None)
BEGIN	Begin work.	date	Decimal
		time	Decimal
		SID	Decimal
		user	ASCII
BEGPREP	Written by the coordinator database server to record the start of the two-phase commit protocol.	flags	Decimal (Value is 0 in a distributed transaction.)
		number of participants	Decimal
BEGWORK	Begin a transaction.	begin transaction time	Decimal
		user ID	Decimal
		process ID	Decimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
BFRMAP	Simple-large-object free-map change.	tblspace ID	Hexadecimal
		bpageno	Hexadecimal
		status	USED/FREE
		log ID	Decimal
		prev page	Hexadecimal
BLDCL	Build tblspace.	tblspace ID	Hexadecimal
		textsize	Decimal
		nextsize	Decimal
		row size	Decimal
		ncolumns	Decimal
		table name	ASCII
BMAPFULL	Bitmap modified to prepare for alter.	tblspace ID	Hexadecimal
		bitmap page num	Decimal
BMAP2TO4	2-bit bitmap altered to two 4-bit bitmaps.	tblspace ID	Hexadecimal
		2-bit bitmap page number	Decimal
		flags	Decimal
BSPADD	Add blobspace.	blobspace name	ASCII
BTCPYBCK	Copy back child key to parent.	tblspace ID	Hexadecimal
		parent logical page	Decimal
		child logical page	Decimal
		slot	Decimal
		rowoff	Decimal
		key number	Decimal
BTMERGE	Merge B-tree nodes.	tblspace ID	Hexadecimal
		parent logical page	Decimal
		left logical page	Decimal
		right logical page	Decimal
		left slot	Decimal
		left rowoff	Decimal
		right slot	Decimal
		right rowoff	Decimal
		key number	Decimal
BTSHUFFL	Shuffle B-tree nodes.	tblspace ID	Hexadecimal
		parent logical page	Decimal
		left logical page	Decimal
		right logical page	Decimal
		left slot	Decimal
		left rowoff	Decimal
		key number	Decimal
		flags	Hexadecimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
BTSPLIT	Split B-tree node.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		parent logical page	Decimal
		left logical page	Decimal
		right logical page	Decimal
		infinity logical page	Decimal
		rootleft logical page	Decimal
		midsplit	Decimal
		key number	Decimal
		key length	Decimal
CDINDEX	Create detached index.	database name	ASCII
		owner	ASCII
		table name	ASCII
		index name	ASCII
CDR	Captures the set of table columns modified by an update statement such as a <i>bitvector</i> . This log record allows Enterprise Replication to capture only the changed data to avoid transmitting the unchanged columns to a target site. In the example, the first six columns of the table are unchanged (6 leftmost bits in the bitvector are 0), the seventh and eighth columns have been updated (seventh and eighth bits are 1), and so on. The onlog output displays as many bits of bitvector as fit in a single line of the output. To see the entire bitvector displayed in hexadecimal, use the onlog -l command.	name of CDR record	ASCII
		partition number	Hexadecimal
		bitvector	Binary
	Sample onlog output for CDR log record: adr len type xid id link name partno bitvector 40 36 CDR 14 0 18 UPDCOLS 10009a 000000110100110100		
CHALLOC	Chunk extent allocation.	pageno	Hexadecimal
		size	Hexadecimal
CHCOMBINE	Chunk extent combine.	pageno	Hexadecimal
CHFREE	Chunk extent free.	pageno	Hexadecimal
		size	Hexadecimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
CHKADJUP	Update chunk adjunct on disk. The database server writes this record when it moves space from the reserved area to the metadata or user-data area or when the user adds an sbspace chunk.	chunk number	Integer
		ud1_start_page	Integer
		ud1_size	Integer
		md_start_page	Integer
		md_size	Integer
		ud2_start_page	Integer
		ud2_size	Integer
		flags	Hexadecimal
CHPHYLOG	Change physical-log location.	pageno	Hexadecimal
		size in kilobytes	Hexadecimal
		dbspace name	ASCII
CHRESERV	Reserve extent for metadata stealing. This record is written when you add an sbspace chunk.	chunk number	Integer
		page number	Integer
		length	Integer
CHSPLIT	Chunk extent split.	pageno	Hexadecimal
CINDEX	Create index.	tblspace ID	Hexadecimal
		low rowid	Decimal
		high rowid	Decimal
		index descriptor	ASCII
COARSELOCK	Coarse-grain locking	tblspace ID	Hexadecimal
		old coarse-locking flag value	Decimal
		new coarse-locking flag value	Decimal
CKPOINT	Checkpoint.	max users	Decimal
		number of active transactions	Decimal
CLR	Compensation-log record; created during a rollback.	(None)	(None)
CLUSIDX	Create clustered index.	tblspace ID	Hexadecimal
		key number	Decimal
COLREPAI	Adjust BYTE, TEXT, or VARCHAR column.	tblspace ID	Hexadecimal
		number of columns adjusted	Decimal
COMMIT	Commit work.	date	Decimal
		time	Decimal
COMTAB	Compact slot table on a page.	logical page number	Decimal
		number slots moved	Decimal
		compressed slot pairs	ASCII
COMWORK	End a transaction and commit work.	end transaction time	Decimal
		begin transaction time	Decimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
DELETE	Delete before-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
DELITEM	Delete item from index.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		logical page	Decimal
		key number	Decimal
		key length	Decimal
DERASE	Drop tblspace in down dbspace.	tblspace number	Hexadecimal
		table lock number	Decimal
DFADDEXT	New extent is added.	partnum	Hexadecimal
		offset of extent entry in list	Hexadecimal
		extent size in pages	Decimal
		physical address of extent	Offset and chunk no-hex
DFDRPEXT	Drop the original extent.	partnum	Hexadecimal
		offset of extent entry in list	Hexadecimal
		original size of this extent	Decimal
		physical address	offset and chunk no-hex
DFEND	End of defragment operation.	partnum	Hexadecimal
DFMVPG	Move page from old extent to new extent.	partnum	Hexadecimal
		offset of new extent	Hexadecimal
		logical page number of source	Hexadecimal
		physical address of destination	Offset and chunk no-hex
		physical address of source	Offset and chunk no-hex
DFREMDUM	Remove the dummy entries.	partnum	Hexadecimal
DFSTART	Start of defragment operation.	partnum	Hexadecimal
DINDEX	Drop index.	tblspace ID	Hexadecimal
		key number	Decimal
DRPBSP	Drop blob space.	blob space name	ASCII
DRPCHK	Drop chunk.	chunk number	Decimal
		chunk name	ASCII
DRPDBS	Drop dbspace.	dbspace name	ASCII
DRPLOG	Drop log.	log number	Decimal
		log size (pages)	Decimal
		pageno	Hexadecimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
ENDTRANS	<p>Written by both the coordinator and participant database servers to record the end of the transaction. ENDTRANS instructs the database server to remove the transaction entry from its shared-memory transaction table and close the transaction.</p> <p>In the coordinator logical log, each BEGPREP that results in a committed transaction is paired with an ENDTRANS record. If the final decision of the coordinator is to roll back the transaction, no ENDTRANS record is written.</p> <p>In the participant logical log, each ENDTRANS record is paired with a corresponding HEURTX record.</p>	(None)	(None)
ERASE	Drop tblspace.	tblspace ID	Hexadecimal
FREE_RE	Allocate extent from reserve extent to metadata or user-data area of an sbpace chunk.	chunk number	Integer
		page number	Integer
		length	Integer
		flag	Hexadecimal
HDELETE	Delete home row.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
HEURTX	Written by a participant database server to record a heuristic decision to roll back the transaction. It should be associated with a standard ROLLBACK record indicating that the transaction was rolled back.	flag	Hexadecimal (Value is always 1.)
HINSERT	Home row insert.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
HUPAFT	Home row update, after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
HUPBEF	<p>Home row update, before-image.</p> <p>In addition, the flag field of the HUPBEF header may include the following values:</p> <p>LM_PREVLSN Confirms that an LSN exists.</p> <p>LM_FIRSTUPD Confirms that this is the first update for this rowID by this transaction.</p>	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
		LSN (optional)	Decimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
HUPDATE	<p>If the home row update before-images and after-images can both fit into a single page, the database server writes a single HUPDATE record.</p> <p>In addition, the flag field of the HUPDATE log may include the following values:</p> <p>LM_PREVLSN Confirms that an LSN exists.</p> <p>LM_FIRSTUPD Confirms that this is the first update for this rowID by this transaction.</p>	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		forward ptr rowid	Hexadecimal
		old slotlen	Decimal
		new slotlen	Decimal
		number of pieces	Decimal
		LSN (optional)	Decimal
IDXFLAGS	Index flags.	tblspace ID	Hexadecimal
		key number	Hexadecimal
INSERT	Insert after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
ISOSPCOMMIT	Log an isolated save-point commit.	end transaction time	Decimal
		begin transaction time	Decimal
LCKLVL	Locking mode (page or row).	tblspace ID	Hexadecimal
		old lockmode	Hexadecimal
		new lockmode	Hexadecimal
LG_ADDBPOOL	Add a buffer pool online.	page size in bytes	Decimal
		number of buffers in the pool	Decimal
		number of lru queues	Decimal
		percent of lru_max_dirty	Decimal
		percent of lru_min_dirty	Decimal
PTRUNCATE	Identifies an intention to truncate a table. The partitions are marked to be dropped or reused, according to the specified command option.	tblspace ID	Hexadecimal
TRUNCATE	TRUNCATE has freed the extents and the transaction will be committed.	tblspace ID	Hexadecimal
MVIDXND	Index node moved to allow for 2-bit to 4-bit bitmap conversion.	tblspace ID	Hexadecimal
		old page number	Decimal
		new page number	Decimal
		parent page number	Decimal
		parent slot number	Decimal
		parent slot offset	Decimal
		key number	Decimal
PBDELETE	Delete tblspace blobpage.	bpageno	Hexadecimal
		status	USED/FREE
		unique ID	Decimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
PBINSERT	Insert tblspace blobpage.	bpageno	Hexadecimal
		tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
		pbrowid	Hexadecimal
PDINDEX	Predrop index.	tblspace ID	Hexadecimal
PGALTER	Page altered in place.	tblspace ID	Hexadecimal
		physical page number	Hexadecimal
PGMODE	Page mode modified in bitmap.	tblspace ID	Hexadecimal
		logical page number	Decimal
		old mode	Hexadecimal
		new mode	Hexadecimal
PERASE	Preerase old file. Mark a table that is to be dropped. The database server frees the space on the commit.	tblspace ID	Hexadecimal
PNGPALIGN8	Use the pages in this tblspace as generic pages.	None	
PNLOCKID	Change tblspaces lockid.	tblspace ID	Hexadecimal
		old lock ID	Hexadecimal
		new lock ID	Hexadecimal
PNSIZES	Set tblspace extent sizes.	tblspace ID	Hexadecimal
		fextsize	Decimal
		nextsize	Decimal
PREPARE	Written by a participant database server to record the ability of the participant to commit the transaction, if so instructed.	DBSERVERNAME of coordinator	ASCII
PTADESC	Add alter description information.	tblspace ID	Hexadecimal
		physical page number of previous page	Hexadecimal
		logical page number	Decimal
		number of columns added	Decimal
PTALTER	Alter of fragment begun.	tblspace ID	Hexadecimal
		physical page number previous page	Hexadecimal
		logical page number	Decimal
		alter desc page number	Decimal
		num columns added	Decimal
		version of alter	Decimal
		added rowsize	Decimal
PTALTNEWKEYD	Update key descriptors in a tblspace header after an alter table command.	bytes in key descriptor	Decimal
		data in key descriptor	ASCII

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
PTALTOLDKEYD	Update key descriptors after an alter table command.	bytes in key descriptor	Decimal
		data in key descriptor	ASCII
PTCOLUMN	Add special columns to fragment.	tblspace ID	Hexadecimal
		number of columns	Decimal
PTEXTEND	Tblspace extend.	tblspace ID	Hexadecimal
		last logical page	Decimal
		first physical page	Hexadecimal
PTRENAME	Rename table.	tblspace ID	Hexadecimal
		old table name	ASCII
		new table name	ASCII
RDELETE	Remainder page delete.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
		hrowid (optional)	Decimal
		poffset (optional)	Decimal
RENDBS	Rename dbspace.	new dbspace name	ASCII
REVERT	Logs the reversion of a database space to a database space of an earlier version.	type of reversion event	Decimal
		arg1	Decimal
		arg2	Decimal
		arg3	Decimal
RINSERT	Remainder page insert.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
		hrowid (optional)	Decimal
		poffset (optional)	Decimal
ROLLBACK	Rollback work.	date	Decimal
		time	Decimal
ROLWORK	End a transaction and roll back work.	end transaction time	Decimal
		begin transaction time	Decimal
RSVEXTEND	Logs the extension to the reserved pages.	number of pages	Decimal
		physical page number of extent	Hexadecimal
RTREE	Logs inserts and deletions for R-tree index pages. (Other operations on R-tree indexes are physically logged.) The record subtypes are: <ul style="list-style-type: none"> • LEAFINS - insert item in a leaf page • LEAFDEL - delete item from leaf page 	record subtype	ASCII
		[index page rowid	Hexadecimal
		tuple length	Decimal
		base table rowid	Decimal
		base table fragid	Decimal
		delete flag]	Decimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
BRUPAFT	Remainder page update, after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
RUPBEF	Remainder page update, before-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
		hrowid (optional)	Decimal
		poffset (optional)	Decimal
RUPDATE	If the remainder page update before-images and after-images can both fit into a single page, the database server writes a single RUPDATE record.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		forward ptr rowid	Hexadecimal
		old slotlen	Decimal
		new slotlen	Decimal
		number of pieces	Decimal
		hrowid (optional)	Decimal
SBLOB	<p>Indicates a subsystem log record for a smart large object.</p> <p>The various record subtypes are:</p> <p>CHALLOC CHCOMBINE CHFREE CHSPLIT CREATE DELETES EXTEND HDRUPD PDELETE PTRUNC REFCOUNT UDINSERT UDINSERT_LT UDUPAFT UDUPAFT_LT UDUPAFT UDUPAFT_LT UDWRITE UDWRITE_LT</p>	Varies	Varies
		For more information, see “Log Record Types for Smart Large Objects” on page 5-14.	
SYNC	Written to a logical-log file if that log file is empty and administrator instructs the database server to switch to next log file.	(None)	(None)
TABLOCKS	Written by either a coordinator or a participant database server. It is associated with either a BEGPREP or a PREPARE record and contains a list of the locked tblspaces (by tblspace number) held by the transaction. (In a distributed transaction, transactions are shown as the owners of locks.)	number of locks	Decimal
		tblspace number	Hexadecimal

Table 5-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
UDINSERT	Append new user data.	chunk	Decimal
		page within chunk	Hexadecimal
		offset within page	Hexadecimal
		data length	Hexadecimal
UDUPAFT	Update user data after-image if a UDWRITE is too expensive.	chunk	Decimal
		page within chunk	Hexadecimal
		offset within page	Hexadecimal
		data length	Hexadecimal
UDUPBEF	Update user-data before-image if a UDWRITE is too expensive.	chunk	Decimal
		page within chunk	Hexadecimal
		offset within page	Hexadecimal
		data length	Hexadecimal
UDWRITE	Update user data (difference image).	chunk	Decimal
		page within chunk	Hexadecimal
		offset within chunk	Hexadecimal
		length before write	Hexadecimal
		length after write	Hexadecimal
UNDO	Header record to a series of transactions to be rolled back.	count	Decimal
UNDOBLDC	This record is written if a CREATE TABLE statement should be rolled back but cannot be because the relevant chunk is down. When the log file is replayed, the table will be dropped.	tblspace number	Hexadecimal
UNIQID	Logged when a new SERIAL value is assigned to a row.	tblspace ID	Hexadecimal
		unique ID	Decimal
UNIQ8ID	Logged when a new SERIAL8 value is assigned to a row.	tblspace ID	Hexadecimal
		unique ID	Decimal
UPDAFT	Update after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
UPDBEF	Update before-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
XAPREPARE	Participant can commit this XA transaction.	(None)	(None)

Log Record Types for Smart Large Objects

All smart-large-object log records are the SBLOB type. Each smart-large-object log record contains six header columns, described in “Logical-Log Record Header” on page 5-2; the record subtype; and additional information. The information that appears varies, depending on record subtype.

Table 5-4 lists all the smart-large-object record types. The **Subtype** column describes the smart-large-object record type. The **Action** column indicates the type of database server action that generated the log entry. The **Additional Columns** and **Format** columns describe what information appears for each record type.

Table 5-4. Record Subtypes for Smart Large Objects

Record Subtype	Action	Additional Columns	Format
CHALLOC	Allocate chunk extent.	extent [chk, page, len]	Decimal
		flags	Hexadecimal
CHCOMBINE	Combine two pages in the user-data extent list.	chunk number	Decimal
		first page	Decimal
		second page	Decimal
CHFREE	Frees chunk extent.	extent [chk, page, len]	Decimal
CHSPLIT	Split a page in the user-data extent list.	chunk number	Decimal
		UDFET page to split	Decimal
CREATE	Create smart large object.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		number of extents in lomaphdr	Decimal
DELETE	Delete a smart large object at commit.	smart-large-object ID [sbs, chk, page, oid]	Decimal
EXTEND	Add extent to an extent list of a smart large object.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		extent [chk, page, len]	Decimal
		lomap overflow page number	Decimal
HDRUPD	Update smart-large-object header page.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		old EOF offset	String
		new EOF offset	String
		old times	Decimal
		new times	Decimal
PDELETE	Queue a smart large object for deletion at commit.	smart-large-object ID [sbs, chk, page, oid]	Decimal
PTRUNC	Queue a smart large object for truncation at commit.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		old offset	String
		new offset	String
REFCOUNT	Increment or decrement the reference count of a smart large object.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		1 if increment; 0 if decrement	Decimal
UDINSERT,	Append new user data.	chunk	Decimal
UDINSERT_LT		page within chunk	Decimal
		offset within page	Decimal
		data length	Decimal

Table 5-4. Record Subtypes for Smart Large Objects (continued)

Record Subtype	Action	Additional Columns	Format
UDUPAFT,	Update user-data after-image if a UDWRITE is too expensive.	chunk	Decimal
UDUPAFT_LT		page within chunk	Decimal
		offset within page	Decimal
		data length	Decimal
UDUPBEF,	Update user-data beforeimage if a UDWRITE is too expensive.	chunk	Decimal
UDUPBEF_LT		page within chunk	Decimal
		offset within page	Decimal
		data length	Decimal
UDWRITE,	Update user data (difference image).	chunk	Decimal
UDWRITE_LT		page within chunk	Decimal
		offset within page	Decimal
		length before write	Decimal
		length after write	Decimal
		number of different image pieces	Decimal

For an example of smart-large-object records in **onlog** output, see smart-large-object log records in the chapter on what is the logical log in the *IBM Informix Administrator's Guide*.

Figure 5-1 shows an example of smart-large-object records in **onlog** output. The first two records show that an extent was freed. The next group of records, flanked by BEGIN and COMMIT, shows the allocation of storage and creation of the smart large objects.

addr	len	type	xid	id	link	subtype	specific-info
4e8428	40	SBLOB	8	0	4e7400	CHFREE	(2,53,421)
4e8450	40	SBLOB	8	0	4e8428	CHFREE	(2,579,421)
c8018	40	BEGIN	8	3	0	07/13/98	10:23:04 34 informix
c8040	264	SBLOB	8	0	c8018	CREATE	[2,2,1,900350517] 10
c8148	44	SBLOB	8	0	c8040	CHALLOC	(2,53,8) 0x1
c8174	68	SBLOB	8	0	c8148	EXTEND	[2,2,1,900350517] (2,53,8) -1
c81b8	264	SBLOB	8	0	c8174	CREATE	[2,2,2,900350518] 10
c82c0	44	SBLOB	8	0	c81b8	CHALLOC	(2,61,1) 0x1
c82ec	68	SBLOB	8	0	c82c0	EXTEND	[2,2,2,900350518] (2,61,1) -1
c8330	56	SBLOB	8	0	c82ec	REFCOUNT	[2,2,1,900350517] 1
c8368	56	SBLOB	8	0	c8330	REFCOUNT	[2,2,2,900350518] 1
c83a0	36	COMMIT	8	0	c8368	07/13/98	10:23:05
c83c4	40	BEGIN	8	3	0	07/13/98	10:23:05 34 informix
c83ec	264	SBLOB	8	0	c83c4	CREATE	[2,2,3,900350519] 10
c84f4	44	SBLOB	8	0	c83ec	CHALLOC	(2,62,1) 0x1
c8520	68	SBLOB	8	0	c84f4	EXTEND	[2,2,3,900350519] (2,62,1) -1
c8564	56	SBLOB	8	0	c8520	REFCOUNT	[2,2,3,900350519] 1
c859c	36	COMMIT	8	0	c8564	07/13/98	10:23:05

Figure 5-1. Smart-Large-Object Records in onlog Output

Part 2. Administrative Utilities

Chapter 6. Overview of Utilities

The Informix database server utilities allow you to perform administrative tasks directly from the command line.

For a complete list of server utilities, see <http://publib.boulder.ibm.com/infocenter/idshelp/v117/topic/com.ibm.sa.doc/utilities.htm>.

The database server utilities support multibyte command-line arguments. For a complete list of the utilities that support multibyte command-line arguments, see the Locale-specific support for utilities (GLS User's Guide).

The database server must be online before you execute a utility, with the following exceptions:

- **oninit**
- Some **onlog** options
- Some **oncheck** options

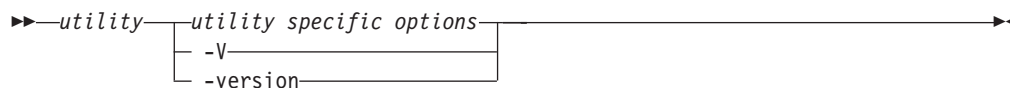
Note: When using utilities, do not use the UNIX command CTRL-C to send an interrupt signal to a process because it might produce an error.

Obtaining Utility Version Information

Many Informix command-line utilities allow you to obtain version information using **-V** and **-version** options. You use the **-V** and **-version** options primarily for debugging. When a Technical Support representative asks for the version number, you can use the **-V** and **-version** options to find the information.

The **-V** option displays the software version number and the serial number.

The **-version** option extends the **-V** option to display additional information on the build operation system, build number, and build date.



The **-V** and **-version** options cannot be used with any other utility options. For example, the **onstat -version** command might display the following output.

```
onstat -version
```

```
Program:          onstat
Build Version:    11.70.FC1
Build Host:       connla
Build OS:         SunOS 5.6
Build Number:     009
Build Date:       Sat Nov 20 03:38:27 CDT 2010
GLS Version:      glslib-4.50.xc2
```

The **onstat -V** command might display the following information:

```
IBM Informix Version 11.70.FC1   Software Serial Number
RDS#N0000000
```

IBM Informix Server Administrator

IBM Informix Server Administrator (ISA) allows a DBA to manage Informix database servers by executing Informix commands from any web browser. You do not need to be familiar with the syntax and format of database server commands. ISA presents the command output in an easy-to-read format.

ISA is available for download at <http://www.ibm.com/software/data/informix/downloads.html>.

With ISA, you can perform these database server administrative tasks:

- Change configuration parameters temporarily or permanently.
- Use **Server Setup** to configure or reconfigure the database server.
- Change the database server mode.
- Modify connectivity information in the **sqlhosts** file.
- Check dbspaces, blobspaces, and sbspaces.
- Manage logical logs and physical logs.
- Examine and modify memory usage.
- Read the message log.
- Back up and restore dbspaces, blobspaces, and sbspaces.
- Run various **onstat** commands to monitor performance.
- Enter SQL statements and examine database schemas.
- Add and remove chunks, dbspaces, blobspaces, sbspaces.
- Examine and manage user sessions.
- Examine and manage virtual processors (VPs).
- Use the High-Performance Loader (HPL), **dbimport**, and **dbexport**.
- Manage Enterprise Replication.
- Manage an Informix MaxConnect server.
- Set up primary and secondary database servers for High-Availability Data Replication.
- Use the following utilities: **dbaccess**, **dbschema**, **onbar**, **oncheck**, **ondblog**, **oninit**, **onlog**, **onmode**, **onparams**, **onspaces**, **onstat**, **onpladm**.
- Enter any Informix utility, UNIX shell command, or Windows command.

Chapter 7. The **genoncfg** Utility

Use the **genoncfg** utility to expedite the process of customizing the default Informix configuration file (**onconfig.std**) to the host environment and your planned usage of a database server instance.

Syntax

```
➤ genoncfg input_file [informixdir]
           -h
           -V
           -version
```

Element	Purpose	Key Considerations
<i>input_file</i>	Name of the input file containing your parameter settings.	
<i>informixdir</i>	Path to the Informix installation that you want to configure.	You can omit the installation path if the INFORMIXDIR environment variable is set. If the INFORMIXDIR variable is already set and you enter an installation path on the command line, the utility runs with the command-line path.
-h	Help information about the genoncfg utility.	
-V	Displays short version information and exits the command-line utility.	
-version	Displays extended version information and exits the command-line utility.	

Usage

Log in to the host computer as root or user **informix** before you run this utility.

You must set parameters that are valid for your host environment in an input file before you can successfully run the **genoncfg** utility. For all environments, the parameter **disk** is required in the input file. You can also enter directives in the input file. The directives are not required to run the utility, but they can be helpful in some circumstances.

The utility does not read or modify any existing configuration file. If you have a pre-existing **ONCONFIG** file in the host environment, none of its parameter values are changed when you run the utility. Therefore, you can review the recommended configuration settings before you put them in effect on a database server instance.

To use the **genoncfg** utility:

1. Create the input file containing your values for the parameters that the **genoncfg** utility processes with a text editor.
2. Run the utility with your input file. The configuration file (named **onconfig**) is generated and saved in the working directory.
3. *Optional:* Rename the generated configuration file.

4. If you want to run a database server instance with the generated configuration file, copy the file to \$INFORMIXDIR/etc and update the **ONCONFIG** environment variable accordingly.

Input File for the **genoncfg** Utility

Use the input file to specify the following information about the database server instance:

- number of anticipated online transaction processing (OLTP) connections
- number of anticipated decision-support systems (DSS) connections
- disk space
- CPU utilization
- network connection settings
- recovery time

The input file is an ASCII text file. There is no required order for the parameters. The following is an example of an input file:

```
cpus 1
memory 1024 m
connection name demo_on onsoctcp 9088
servernum 1
oltp_connections 10
dss_connections 2
disk /opt/IBM/informix/demo/server/online_root 0 k 300 m
directive one_crit
directive debug
```

*Table 7-1. Parameters of the Input File for the **genoncfg** Utility*

Parameter	Description
connection	<p>Server connection parameters:</p> <ul style="list-style-type: none">• name or alias, depending on whether the connection functions with a specific server name (the DBSERVERNAME parameter of the configuration file) or with an alternative server name (using the DBSERVERALIASES parameter of the configuration file)• name for the connection• type of server connection (equivalent to NETTYPE in the configuration file)• port number for the service <p>Example: connection name demo_on onsoctcp 9088</p>
cpus	<p>Number of central processing units (CPUs) to allocate the instance. Example: cpus 1</p>

Table 7-1. Parameters of the Input File for the **genoncfg** Utility (continued)

Parameter	Description
directive	<p>Directives that can be used with the genoncfg utility.</p> <ul style="list-style-type: none"> • one_crit: Configures the database server to store physical logs, logical logs, and data in the root dbspace only. • debug: Displays information in real time about the host environment and actions done on the configuration file. <p>Example: directive one_crit</p> <p>This information can be helpful in troubleshooting problems with database server configuration. One scenario is that the debug directive can result in saving time. In this scenario, you read the displayed information and notice that the utility is creating an onconfig file that you do not want or that will not function. You stop the utility while it is still running, adjust the input file settings, and then rerun the utility with the modified input file.</p>
disk	<p>Disk storage space settings for the instance:</p> <ul style="list-style-type: none"> • location of the root dbspace • size of offset, in megabytes (m) or kilobytes (k) • size of root dbspace, in megabytes (m) or kilobytes (k) <p>Example:</p> <p>UNIX: /opt/IBM/dbspace/rootdbs</p> <p>Windows: d:\INFXDATA\rootdbs</p> <p>Important: If you enter a path location that is the root dbspace of a working instance, the instance is overwritten and made unusable.</p>
dss_connections	<p>Estimated number of decision-support systems (DSS) connections to the instance. For example, a query client or other application that obtains result sets for business intelligence can be a DSS connection. Example: dss_connections 2</p>
memory	<p>Amount of memory, in megabytes (m), for the instance. Example: memory 1024 m</p>
oltp_connections	<p>Estimated number of online transaction processing (OLTP) connections to the instance. Typically, an application that modifies the state of databases in the instance is an OLTP connection. Example: oltp_connections 10</p>

Table 7-1. Parameters of the Input File for the **genoncfg** Utility (continued)

Parameter	Description
rto_server_restart	Specifies the amount of time, in seconds, that the database server has to recover from a problem after you restart Informix and bring it into online or quiescent mode. The value can be set either to 0 to disable the configuration parameter or to a value between 60 and 1800 to enable the parameter and indicate the number of seconds. Example: rto_server_restart 100 specifies the recovery time objective as 100 seconds.
servernum	Unique ID of the database server instance. Example: servernum 1

Chapter 8. The oncheck Utility

Use the **oncheck** utility to check specified disk structures for inconsistencies, repair inconsistent index structures, and display information about disk structures.

The **oncheck** utility requires sort space when examining an index. The amount of sort space required is the same as that needed to build the index. For information about calculating the amount of temporary space needed, see Estimating temporary space for index builds (Performance Guide). If you receive the error "no free disk space for sort," you must estimate the amount of temporary space needed and make that space available.

You can use SQL administration API commands that are equivalent to some **oncheck** commands.

oncheck Check-and-Repair

The **oncheck** utility can repair the following types of disk structures:

- Partition page statistics
- Bitmap pages
- Partition blobpages
- Blobspace blobpages
- Indexes
- Sbspace pages
- Metadata partitions for sbspaces

If **oncheck** detects inconsistencies in other structures, messages alert you to these inconsistencies, but **oncheck** cannot resolve the problem. For more information, see the chapter on consistency checking in the *IBM Informix Administrator's Guide* and Chapter 4, "Disk Structures and Storage," on page 4-1.

What Does Each Option Do?

As Table 8-1 on page 8-2 shows, the **oncheck** options fall into three categories: check, repair, and display. The display or print options (those prefixed with the letter **p**) are identical in function to the **-c** options, except that the **-p** options display additional information about the data that is being checked as the **oncheck** utility executes. You cannot combine **oncheck** option flags except as the following paragraphs describe.

In general, the **-c** options check for consistency and display a message on the screen only if they find an error or inconsistency.

Any user can execute the check options. On UNIX platforms, you must be user **informix** or **root** to display database data or initiate repair options. On Windows, you must be a member of the **Informix-Admin** group to display database data or initiate repair options.

Table 8-1 associates **oncheck** options with their function. It also shows the SQL administration API *command* strings that are equivalent to the **oncheck -c** options.

Table 8-1. oncheck Options and Their Function

Object	Check	SQL administration API <i>command</i> string	Repair	Display
Blobspace simple large objects				-pB
System catalog tables	-cc			-pc
Data rows, no simple large objects or smart large objects	-cd			-pd
Data rows, simple large objects but no smart large objects	-cD			-pD
Table with a user-defined access method	-cd, -cD	CHECK DATA		
Chunks and extents	-ce	CHECK EXTENTS		-pe
Index (key values)	-ci, -cix		-ci -y -pk -y, -pkx -y	-pk
Index (keys plus rowids)	-cI, -cIx		-cI -y -pK -y, -pKx -y	-pK
Index with a user-defined access method	-ci, -cI			
Index (leaf key values)			-pl -y, -plx -y	-pl
Index (leaf keys plus rowids)			-pL -y, -pLx -y	-pL
Pages (by table or fragment)				-pp
Pages (by chunk)				-pP
Root reserved pages	-cr, -cR			-pr, -pR
Metadata for smart large objects	-cs, -cS			-ps, -pS
Space usage (by table or fragment)		CHECK PARTITION PRINT PARTITION		-pt
Space usage (by table, with indexes)				-pT

Using the -y Option to Perform Repairs

Use the **-y** option to instruct **oncheck** to perform repairs automatically, as the following examples show:

```
oncheck -cd -y
oncheck -cD -y
oncheck -ci -y
oncheck -cI -y
```

If you do not use the **-y** option, **oncheck** prompts you when it encounters an inconsistency and allows you to request a repair. If you specify option **-n**, **oncheck** does not prompt you because this option instructs **oncheck** to not perform repairs.

Repairing Indexes in Sbspaces and External Spaces

The **oncheck** utility can repair an index in an sbspaces or external space if the index is created using an access method that supports the **oncheck -y** option. Although the **oncheck** utility does not repair fragmented indexes, user-defined access methods can repair them. For more information about the **oncheck** options that access methods support, see the *IBM Informix DataBlade API Programmer's Guide* or the *IBM Informix Virtual-Index Interface Programmer's Guide*.

Locking and oncheck

The **oncheck** utility places a shared lock on a table during the following operations, so no other users can perform updates, inserts, or deletes until the check has completed:

- When it checks data
- When it checks indexes (with **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL**) and the table uses page locking
- When you specify the **-x** option with **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** and the table uses row locking

If the table does not use page locking, the database server does not place a shared lock on the table when you check an index with the **oncheck -ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** options. When no shared lock is on the table during an index check, other users can update rows during the check.

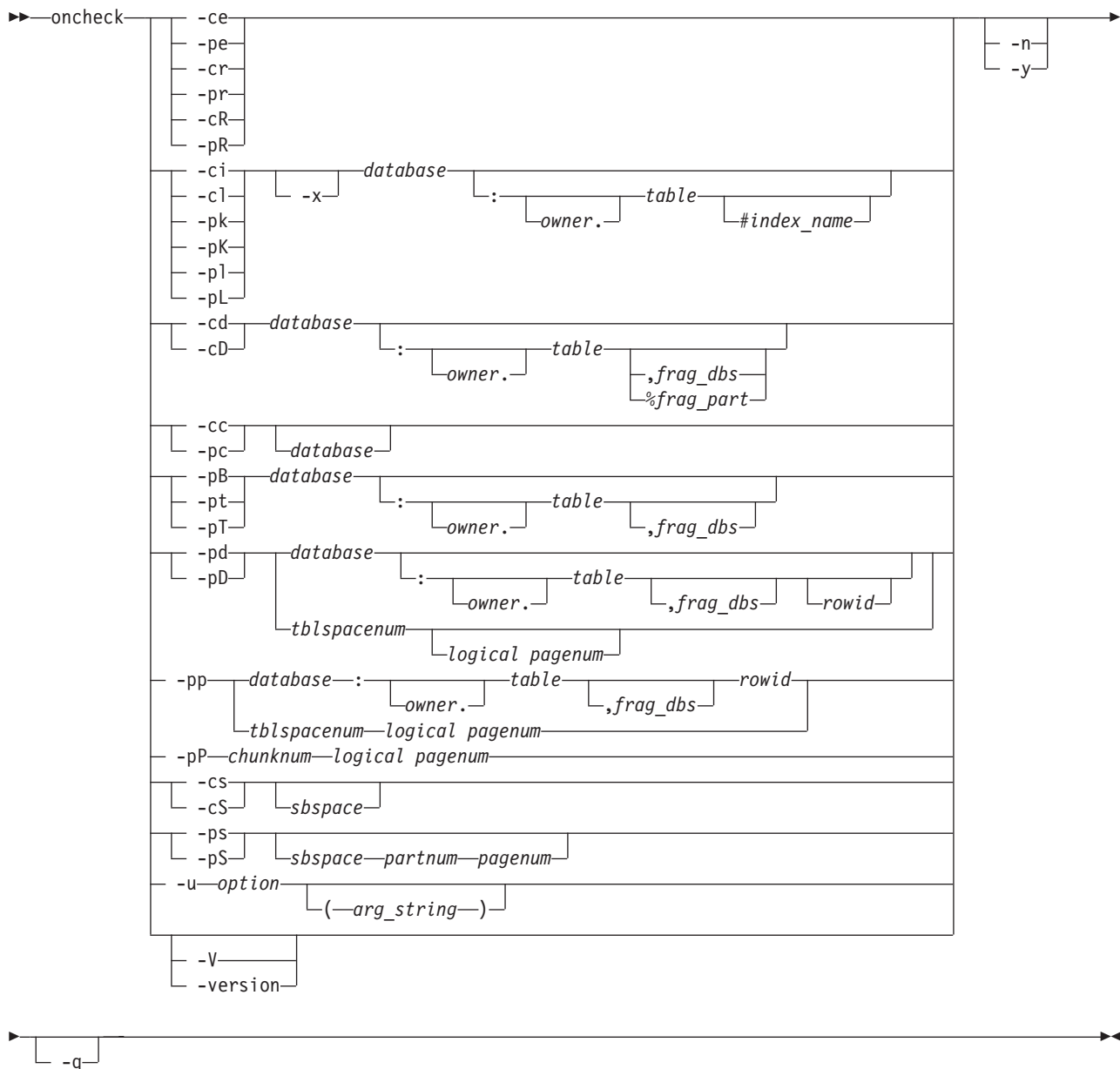
By not placing a shared lock on tables using row locks during index checks, the **oncheck** utility cannot be as accurate in the index check. For absolute assurance of a complete index check, you can execute **oncheck** with the **-x** option. With the **-x** option, **oncheck** places a shared lock on the table, and no other users can perform updates, inserts, or deletes until the check has completed.

The **oncheck** utility returns unreliable results when run on secondary servers in a high-availability cluster.

For more information about the **-x** option, refer to “Turn On Locking with -x” on page 8-21. For information on shared locks and intent shared locks, see the *IBM Informix Performance Guide*.

The **oncheck** utility places a shared lock on system catalog tables when they are checked. It places an exclusive lock on a table when it executes repair options.

oncheck syntax



Element	Purpose	Key Considerations
-cc	Checks system catalog tables for the specified database	See “oncheck -cc and -pc: Check system catalog tables” on page 8-8.
-cd	Reads all pages except simple large objects from the tblspace for the specified database, table, or fragment and checks each page for consistency Also checks tables that use a user-defined access method	Does not check simple or smart large objects. See “oncheck -cd and oncheck -cD commands: Check pages” on page 8-8.
-cD	Same as -cd but also reads the header of each blobpage and checks it for consistency	Checks simple large objects but not smart large objects. See “oncheck -cd and oncheck -cD commands: Check pages” on page 8-8.

Element	Purpose	Key Considerations
-ce	Checks each chunk-free list and corresponding free space and each tblspace extent. Also checks smart-large-object extents and sbspace metadata	The oncheck process verifies that the extents on disk correspond to the current control information that describes them. See “oncheck -ce, -pe: Check the chunk-free list” on page 8-10. For background information, see “Next-Extent Allocation” on page 4-10.
-ci	Checks the ordering of key values and the consistency of horizontal and vertical node links for all indexes associated with the specified table Also checks indexes that use a user-defined access method	See “ oncheck -ci and -cI : Check index node links” on page 8-10.
-cI	Same as -ci but also checks that the key value tied to a rowid in an index is the same as the key value in the row	See “ oncheck -ci and -cI : Check index node links” on page 8-10.
-cr	Checks each of the root dbspace reserved pages for several conditions	See “oncheck -cr and -cR: Check reserved pages” on page 8-12.
-cR	Checks the root dbspace reserved pages, physical-log pages, and logical-log pages	See “oncheck -cr and -cR: Check reserved pages” on page 8-12
-cs	Checks smart large object and sbspace metadata for an sbspace	See “oncheck -cs, -cS, -ps, -pS: Check and display sbspaces” on page 8-13.
-cS	Checks smart large object and sbspace metadata for an sbspace as well as extents	See “oncheck -cs, -cS, -ps, -pS: Check and display sbspaces” on page 8-13.
sbspace	Indicates optional sbspace name If not supplied, all sbspaces are checked.	None.
-n	Indicates that no index repair should be performed, even if errors are detected	Use with the index repair options (-ci , -cI , -pk , -pK , -pl , and -pL).
-pB	Displays statistics that describe the average fullness of blobspace blobpages in a specified table	These statistics provide a measure of storage efficiency for individual simple large objects in a database or table. If a table or fragment is not specified, statistics are displayed for the entire database. See “oncheck -pB: Display blobspace statistics” on page 8-13. For information about optimizing blobspace blobpage size, see the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-pc	Same as -cc but also displays the system catalog information as it checks the system catalog tables, including extent use for each table	None.
-pd	Displays rows in hexadecimal format	See “oncheck -pd and pD: Display rows in hexadecimal format” on page 8-13.

Element	Purpose	Key Considerations
-pD	Displays rows in hexadecimal format and simple-large-object values stored in the tblspace or header information for smart large objects stored in an sbspace sbpage and simple large objects stored in a blobSPACE blobpage	See “oncheck -pd and pD: Display rows in hexadecimal format” on page 8-13.
-pe	Same as -ce but also displays the chunk and tblspace extent information as it checks the chunk free list, the corresponding free space, and each tblspace extent	See “oncheck -ce, -pe: Check the chunk-free list” on page 8-10.
-pk	Same as -ci but also displays the key values for all indexes on the specified table as it checks them	See “oncheck -pk, -pK, -pl, -pL: Display index information” on page 8-15.
-pK	Same as -ci but also displays the key values and rowids as it checks them	See “oncheck -pk, -pK, -pl, -pL: Display index information” on page 8-15.
-pl	Same as -ci but also displays the key values. Only leaf-node index pages are checked	See “oncheck -pk, -pK, -pl, -pL: Display index information” on page 8-15.
-pL	Same as -ci but also displays the key values and rowids for leaf-node index pages only	See “oncheck -pk, -pK, -pl, -pL: Display index information” on page 8-15.
-pp	Displays contents of a logical page	See “oncheck -pp and -pP: Display the contents of a logical page” on page 8-16.
-pP	Same as -pp but requires a chunk number and logical page number or internal rowid as input	See “oncheck -pp and -pP: Display the contents of a logical page” on page 8-16.
-pr	Same as -cr but also displays the reserved-page information as it checks the reserved pages	See “oncheck -pr and pR: Display reserved-page information” on page 8-18.
-pR	Same as -cr but also displays the information for the reserved pages, physical-log pages, and logical-log pages	See “oncheck -pr and pR: Display reserved-page information” on page 8-18.
-ps	Checks and displays smart-large-object and sbspace metadata for an sbspace	See “oncheck -cs, -cS, -ps, -pS: Check and display sbspaces” on page 8-13.
-pS	Checks and displays smart-large-object and sbspace metadata. Lists extents and header information for individual smart large objects	See “oncheck -cs, -cS, -ps, -pS: Check and display sbspaces” on page 8-13.
-pt	Displays tblspace information for a table or fragment	See “oncheck -pt and -pT: Display tblspaces for a Table or Fragment” on page 8-19.
-pT	Same as -pt but also displays index-specific information and page-allocation information by page type (for dbspaces)	See “oncheck -pt and -pT: Display tblspaces for a Table or Fragment” on page 8-19.
-q	Suppresses all checking and validation message	None.
-x	Places a shared lock on the table when you check and print an index	Use with the -ci , -ci , -pk , -pK , -pl , or -pL options. For complete information, see “Turn On Locking with -x” on page 8-21.

Element	Purpose	Key Considerations
-y	Repairs indexes when errors are detected	None.
-V	Displays the software version number and the serial number	See “Obtaining Utility Version Information” on page 6-1.
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See “Obtaining Utility Version Information” on page 6-1.
<i>chunknum</i>	Specifies a decimal value that you use to indicate a particular chunk	Value must be an unsigned integer greater than 0. Chunk must exist. Execute the -pe option to learn which chunk numbers are associated with specific dbspaces, blobspaces or sbspaces.
<i>database</i>	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
db1	Specifies the local database that contains a data type that you want to check	Optionally specify the local database server name using the format db1@server1 .
db2	Specifies the remote database that contains a data type that you want to check	Optionally specify the remote database server name using the format db2@server2 .
<i>frag_dbs</i>	Specifies the name of a dbspace that contains a fragment you want to check for consistency	Dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>index_name</i>	Specifies the name of the index that you want to check for consistency	Index must exist on table and in database specified. Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>logical pagenum</i>	Specifies an integer value that you use to indicate a particular page in a tblspace	Value must be an unsigned integer between 0 and 16,777,215, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
<i>object</i>	Specifies the name of the DataBlade, cast, operator , user-defined data type, or UDR that you want to check	If you do not specify an object name, the database server compares all objects of the same type with the same name and owner.
<i>owner</i>	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; for more information, see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>pagenum</i>	Indicates the page number of the sbpace metadata portion to check and display	None.
<i>partnum</i>	Identifies the sbpace metadata partition to check and display	None.
<i>rowid</i>	Identifies the rowid of the row whose contents you want to display. The rowid is displayed as part of oncheck -pD output	Value must be an unsigned integer between 0 and 4,277,659,295, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
<i>sbspace</i>	Specifies the name of the sbpace that you want to check for consistency	None.

Element	Purpose	Key Considerations
<i>server</i>	Specifies the database server name	If you omit the database server name, oncheck uses the name that INFORMIXSERVER specifies.
<i>table</i>	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; for more information, see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>tblspacenum</i>	Identifies the tblspace whose contents you want to display	Value must be an unsigned integer between 0 and 208,666,624, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.

oncheck -cc and -pc: Check system catalog tables

Syntax:

```

>> oncheck [ -cc | -pc ] database

```

The **-cc** option checks all system catalog tables for the specified database. If you do not specify a database, it checks all system catalog tables for all databases.

The **-pc** option performs the same checks on system catalog tables and also displays the system catalog information, including the physical address, type of locking used, row size, number of keys, extent use, the number of pages allocated and used, tblspace partnum, and index use for each table.

Before you execute **oncheck -cc** or **oncheck -pc**, execute the SQL statement UPDATE STATISTICS to ensure that an accurate check occurs. To check a table, **oncheck** compares each system catalog table to its corresponding entry in the tblspace. For more information about the tblspace, see “Structure of the Tblspace Tblspace” on page 4-4.)

oncheck -cd and oncheck -cD commands: Check pages

Use the **oncheck -cd** and **oncheck -cD** commands to check each page for consistency.

Syntax:

```

>> oncheck [ -cd | -cD ] database [ : [ owner. ] table [ , frag_dbs | %frag_part ]

```

The **oncheck -cd** command reads all pages, except for blobpages and sbpages, from the tblspace for the specified database, table, fragment, or multiple fragments (fragparts), and checks each page for consistency. This command compares entries in the bitmap page to the pages to verify mapping.

The **oncheck -cD** command performs the same checks as the **oncheck -cd** command, and also checks the header of each blobpage for consistency. The **oncheck -cD** command does not compare the beginning time stamps stored in the

header with the ending time stamps stored at the end of a blobpage. Use the **oncheck -cd -y** command to clean up orphaned simple large objects in blobspaces, which may occur after a rollback across several log files.

If the database contains fragmented tables, but no fragment is specified, the **oncheck -cd** command checks all fragments in the table. If you do not specify a table, the command checks all of the tables in the database. By comparison, the **oncheck -pd** command displays a hexadecimal dump of specified pages but does not check for consistency.

For both the **oncheck -cd** and **oncheck -cD** commands, the **oncheck** utility locks each table as it checks the indexes for the table. To repair the pages, use **oncheck -cd -y** or **oncheck -cD -y**.

If tables are fragmented on multiple partitions in the same dbspace, the **oncheck -cd** and **oncheck -cD** commands show the partition names. The following example shows typical output for a table that has fragments in multiple partitions in the same dbspace:

```
TBLspace data check for multipart:informix.t1
Table fragment partition part_1 in DBspace dbs1
Table fragment partition part_2 in DBspace dbs1
Table fragment partition part_3 in DBspace dbs1
Table fragment partition part_4 in DBspace dbs1
Table fragment partition part_5 in DBspace dbs1
```

When you use the **oncheck -cd** or **oncheck -cD** command, you can specify either the *frag_dbs* or the *%frag_dbs* option but not both:

- When you use the *frag_dbs* option, the utility checks all fragments in the dbspace *frag_dbs*.
- When you use the *%frag_dbs* option, the utility checks only the fragment named *frag_part*, if the PARTITION syntax was used when the fragment or table was created.

While it is possible to fragment an index with the PARTITION syntax, it is not possible to limit an index check to just one fragment or partition. For example, you can specify **oncheck -cDI my_db:my_tab,data_dbs1** or **oncheck -cDI my_db:my_tab%part1**. The "D" (data) portion of the check is limited according to the specification, however the "I" (index) check is not limited.

Examples

The following example checks the data rows, including simple large objects and smart large objects, in the **catalog** table:

```
oncheck -cD superstores_demo:catalog
```

If you specify a single fragment, the **oncheck** utility displays a single header for that fragment. For fragmented tables, one header is displayed for each fragment:

```
TBLspace data check for stores_demo:informix.tab1
Table fragment in DBspace db1
```

Messages

If the **oncheck** utility finds no inconsistencies, a header displays for each table that the utility. For example:

```
TBLSPACE data check for stores_demo:informix.customer
```

If the **oncheck** utility finds an inconsistency a message displays. For example:
BAD PAGE 2:28: pg_addr 2:28 != bp-> bf_pagenum 2:69

The physical address 2:28 represents page 28 of chunk number 2.

If an index that uses an access method provided by a DataBlade module cannot find the access method, you receive the following message:

-9845 Access method *access_method_name* does not exist in database.
Ensure that the DataBlade installation was successful.

Reference

To monitor blob space blob pages, refer to the **oncheck -pB** command for information at “oncheck -pB: Display blob space statistics” on page 8-13).

Use CHECK DATA as the SQL administration API *command* string for the **oncheck -cd** command.

Related reference:

“check data argument: Check data consistency (SQL administration API)” on page 21-23

oncheck -ce, -pe: Check the chunk-free list

Syntax:

►► oncheck [-ce] [-pe]

The **-ce** option checks each chunk-free list and corresponding free space and each tblspace extent. For more information, refer to “Next-Extent Allocation” on page 4-10 and “Structure of the Chunk Free-List Page” on page 4-4, respectively. The **oncheck** process verifies that the extents on disk correspond to the current control information that describes them.

The **-pe** option performs the same checks and also displays the chunk and tblspace extent information during the check. The **-ce** and **-pe** options also check blob spaces, smart-large-object extents, and user-data and metadata information in sbspace chunks.

For information about using **oncheck -ce** and **-pe**, see managing disk space in the *IBM Informix Administrator's Guide*.

Use CHECK EXTENTS as the SQL administration API *command* string for **oncheck -ce**.

Related reference:

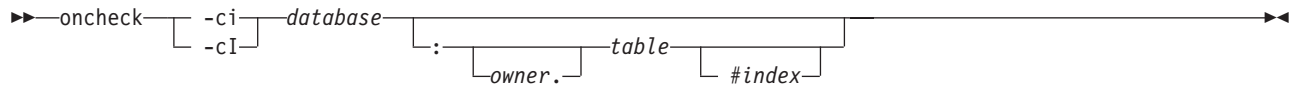
“check extents argument: Check extent consistency (SQL administration API)” on page 21-24

oncheck -ci and -cl: Check index node links

Use the **oncheck -ci** and **oncheck -cl** commands to check the ordering of key values and the consistency of horizontal and vertical node links for all indexes associated with the specified table.

The **oncheck -cI** command also checks that the key value tied to a rowid in an index is the same as the key value in the row. The **-cI** option does not cross-check data on a functional index.

Syntax:



If you do not specify an index, the option checks all indexes. If you do not specify a table, the option checks all tables in the database.

The same **-ci** repair options are available with **-cI**. If **oncheck -ci** or **oncheck -cI** detects inconsistencies, it prompts you for confirmation to repair the problem index. If you specify the **-y** (yes) option, indexes are automatically repaired. If you specify the **-n** (no) option, the problem is reported but not repaired; no prompting occurs.

If **oncheck** does not find inconsistencies, the following message appears:
validating indexes.....

The message displays the names of the indexes that **oncheck** is checking.

Note: Using **oncheck** to rebuild indexes can be time consuming. Processing is usually faster if you use the SQL statements DROP INDEX and CREATE INDEX to drop and re-create the index.

The following example checks all indexes on the **customer** table:

```
oncheck -cI -n stores_demo:customer
```

The following example checks the index **zip_ix** on the **customer** table:

```
oncheck -cI -n stores_demo:customer#zip_ix
```

If indexes are fragmented on multiple partitions in the same dbspace, the **oncheck -ci** and **oncheck -cI** commands show the partition names. The following example shows typical output for an index that has fragments in multiple partitions in the same dbspace:

```
Validating indexes for multipart:informix.tl...
Index idx_t1
Index fragment partition part_1 in DBspace dbs1
Index fragment partition part_2 in DBspace dbs1
Index fragment partition part_3 in DBspace dbs1
Index fragment partition part_4 in DBspace dbs1
Index fragment partition part_5 in DBspace dbs1
```

By default, the database server does not place a shared lock on the table when you check an index with the **oncheck -ci** or **oncheck -cI** commands unless the table uses page locking. For absolute assurance of a complete index check, you can execute **oncheck -cior oncheck -cI** with the **-x** option. With the **-x** option, **oncheck** places a shared lock on the table, and no other users can perform updates, inserts, or deletes until the check has completed. For more information about using **oncheck -ci** and **oncheck -cI** with the **-x** option, "Turn On Locking with -x" on page 8-21.

When you execute **oncheck** on an external index, the user-defined access method is responsible for checking and repairing an index. If an index that employs a user-defined access method cannot find the access method, the database server reports an error. The **oncheck** utility does not repair inconsistencies in external indexes. You should not use **oncheck -cI** on a table that contains more than one type of index.

The **oncheck** utility requires sort space when examining an index. The amount of sort space required is the same as that needed to build the index. For information about calculating the amount of temporary space needed, see Estimating temporary space for index builds (Performance Guide). If you receive the error "no free disk space for sort," you must estimate the amount of temporary space needed and make that space available.

Important: If you are using the Verity Text Search DataBlade Module, the **-cI** option performs an index merge instead of the usual operations.

For more information about indexes, see "Structure of B-Tree Index Pages" on page 4-16.

oncheck -cr and -cR: Check reserved pages

Syntax:

```
►► oncheck [ -cr | -cR ]
```

The **-cr** option checks each of the root dbspace reserved pages as follows:

- It validates the contents of the ONCONFIG file with the PAGE_CONFIG reserved page.
- It ensures that all chunks can be opened, that chunks do not overlap, and that chunk sizes are correct.

The **-cR** option performs the same checking and validation, and also checks all logical-log and physical-log pages for consistency. The **-cr** option is considerably faster because it does not check the log-file pages.

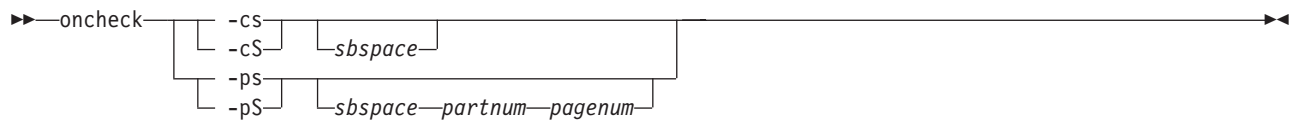
If you have changed the value of a configuration parameter (either through ISA, **onparams**, **onmonitor**, **onspaces**, or by editing the configuration file), but you have not yet reinitialized shared memory, **oncheck -cr** and **oncheck -cR** detect the inconsistency and return an error message.

If **oncheck -cr** does not display any error messages after you execute it, you can assume that all three items in the preceding list were checked successfully.

For more information on reserved pages, see "Reserved Pages" on page 4-2.

oncheck -cs, -cS, -ps, -pS: Check and display sbspaces

Syntax:



The **-cs** option checks sbspaces. The **-ps** option checks sbspaces and extents.

The **-cS** option validates and displays metadata for an sbspace.

The **-ps** option checks sbspaces and extents. If you do not specify the sbspace name, these options check all sbspaces.

The **-pS** option validates and displays metadata for an sbspace and also lists extents and header information for smart large objects.

If you do not specify the sbspace name, all sbspaces will be checked. The following example checks and displays metadata for **test_sbspace**:

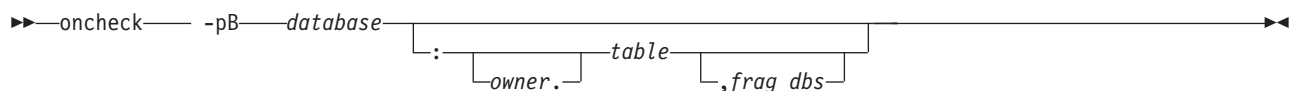
```
oncheck -ps test_sbspace
```

If you specify **rootdbs** as the sbspace name with the **-cs** or **-ps** options, **oncheck** checks the root dbspace.

For more information about using the **-cs**, **-cS**, **-ps**, and **-pS** options, see the *IBM Informix Administrator's Guide*.

oncheck -pB: Display blobspace statistics

Syntax:

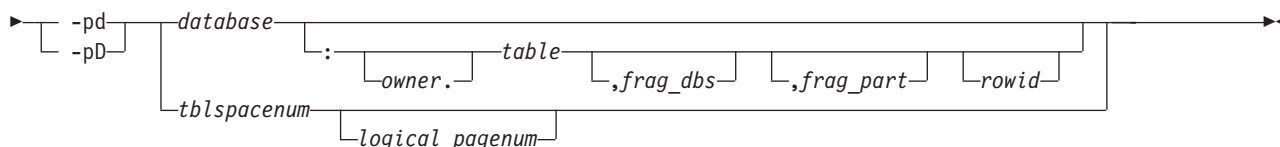


The **-pB** option displays statistics that describe the average fullness of blobpage blobpages in a specified table. These statistics provide a measure of storage efficiency for individual simple large objects in a database or table. If you do not specify a table or fragment, the option displays statistics for the entire database. For more information, see optimizing blobpage blobpage size in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

oncheck -pd and pD: Display rows in hexadecimal format

Syntax:





The **-pd** option takes a database, a table, a fragment, a fragment partition (fragpart), and a specific rowid or tblspace number and logical page number as input. In every case, **-pd** prints page-header information and displays the specified rows for the database object (database, table, fragment, internal rowid, or page number) that you specify in hexadecimal and ASCII format. No checks for consistency are performed.

Element	Purpose	Key Considerations
<i>database</i>	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>frag_dbs</i>	Specifies the name of a dbspace that contains a fragment you want to check for consistency	Dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>frag_part</i>	Specifies the fragment partition	For fragmented tables or an index that use expression-based or round-robin distribution schemes, you can create multiple partitions, which are collections of pages for a table or index, within a single dbspace. This partition is referred to as a <i>fragment partition</i> or <i>fragpart</i> .
<i>logical pagenum</i>	Specifies an integer value that you use to indicate a particular page in a tblspace	Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier. Value must be an unsigned integer between 0 and 16,777,215, inclusive.
<i>owner</i>	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>rowid</i>	Identifies the rowid of the row whose contents you want to display. The rowid is displayed as part of oncheck -pD output	Value must be an unsigned integer between 0 and 4,277,659,295, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
<i>table</i>	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>tblspacenum</i>	Identifies the tblspace whose contents you want to display	Value must be an unsigned integer between 0 and 208,666,624, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.

If you specify an internal rowid (expressed as a hexadecimal value), the rowid maps to a particular page, and all rows from that page are printed.

If you specify a logical page number (expressed as a decimal), all the rows of the tblspace number with the logical page number are printed.

Element	Purpose	Key Considerations
<i>database</i>	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>chunknum</i>	Specifies a decimal value that you use to indicate a particular chunk	Value must be an unsigned integer greater than 0. Chunk must exist.
<i>frag_dbs</i>	Specifies the name of a dbspace that contains a fragment you want to check for consistency	Dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>frag_part</i>	Specifies the partition name of the fragment to be checked. This is useful in cases where more than one fragment of a table was created in the same dbspace.	For fragmented tables or an index that use expression-based or round-robin distribution schemes, you can create multiple partitions, which are collections of pages for a table or index, within a single dbspace. This partition is referred to as a <i>fragment partition</i> or <i>fragpart</i> .
<i>logical pagenum</i>	Specifies an integer value that you use to indicate a particular page in a tblspace	Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier. Value must be an unsigned integer between 0 and 16,777,215, inclusive.
<i>owner</i>	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>rowid</i>	Identifies the rowid of the row whose contents you want to display. The rowid is displayed as part of oncheck -pD output	Value must be an unsigned integer between 0 and 4,277,659,295, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
<i>table</i>	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>tblspacenum</i>	Identifies the tblspace whose contents you want to display	Value must be an unsigned integer between 0 and 208,666,624, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.

The **-pp** option has the following syntax variations:

Invocation	Explanation
oncheck -pp tblspc lpn <pages>	Displays the contents of a logical page using a tblspace number and logical page number. You can also specify an optional parameter specifying the number of pages to be printed.
oncheck -pp tblspc lpn -h	Displays only the header of a logical page using a tblspace number and logical page number.
oncheck -pp database:table rowid	Displays the contents of a logical page using a database name, table name, and an Informix internal rowid. You can obtain this internal rowid with the oncheck -pD command. This internal rowid is not the serial rowid that is assigned in tables created with the CREATE TABLE tablename WITH ROWIDS statement. For more information, see “Definition of Rowid” on page 4-13

invocations of the **oncheck -pp** command:

```
oncheck -pp stores_demo:orders 0x211 # database:owner.table, # fragment rowid
oncheck -pp stores_demo:informix.customer,frag_dbspcel 0x211
oncheck -pp 0x100000a 25 # specify the tblspace number and # logical page number
```

The **-pP** option provides the following syntax variations:

Invocation	Explanation
oncheck -pP chunk# offset pages	Displays the contents of a logical page using a chunk number and an offset. You can also specify an optional parameter specifying the number of pages to be printed.
oncheck -pP <i>chunk# offset -h</i>	Displays only the header of a logical page using a chunk number and an offset.

Note: The output for chunk page displays both the start and the length fields in decimal format.

The following example shows typical output using the **onstat -pP** command:

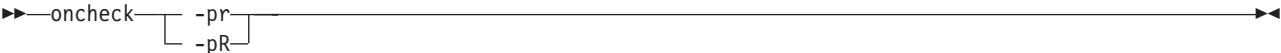
```

oncheck -pP 1 5 2
addr      stamp      nslots      flag      type      frptr      frcnt      next      prev
stamp     100005      250181      2         1000      ROOTRSV      320        1716      0
0         250181      slot      ptr      len      flg
...
addr      stamp      nslots      flag      type      frptr      frcnt      next      prev
stamp     100005      6          250182    2         1000      ROOTRSV      128        1908      0
250182    slot      ptr      len      flg      1         24          56         0
2         80         48         0

```

oncheck -pr and pR: Display reserved-page information

Syntax:



The **-pr** option performs the same checks as **oncheck -cr** and displays the reserved-page information.

The **-pR** option performs the same checks as **oncheck -cR**, displays the reserved-page information, and displays detailed information about logical-log and physical-log pages, marking the start and end of the active physical-log pages.

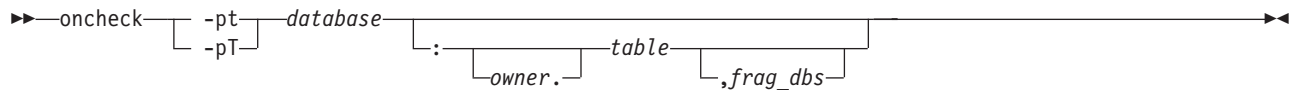
If you have changed the value of a configuration parameter (either through ISA or by editing the configuration file), but you have not yet reinitialized shared memory, **oncheck -pr** and **oncheck -pR** detect the inconsistency and return an error message.

For a listing and explanation of **oncheck -pr** output, see “Reserved Pages” on page 4-2. For a description of the **-cr** option, see “oncheck -cr and -cR: Check reserved pages” on page 8-12.

oncheck -pt and -pT: Display tblspaces for a Table or Fragment

The **oncheck -pt** and **oncheck -pT** options print a tblspace report for a specific table or fragment. The only difference between these options is that **oncheck -pT** prints more information than **oncheck -pt**.

Syntax:



Element	Purpose	Key Considerations
<i>database</i>	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>frag_dbs</i>	Specifies the name of a dbspace that contains a fragment you want to check for consistency	Dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>owner</i>	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; see <i>IBM Informix Guide to SQL: Syntax</i> .
<i>table</i>	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; see <i>IBM Informix Guide to SQL: Syntax</i> .

The **-pt** option prints a tblspace report for the table or fragment with the name and database that you specify when you execute **oncheck** at the command line. The report contains general allocation information including the maximum row size, the number of keys, the number of extents, their sizes, the pages allocated and used per extent, the current serial value, and the date that the table was created. The **-pt** output prints the pagesize of the tblspace, the number of pages (allocated, used and data) in terms of logical pages. The **Extents** fields list the physical address for the tblspace **tblspace** entry for the table and the address of the first page of the first extent. The extent list shows the number of logical as well as physical pages in every extent. If you do not specify a table, the option displays this information for all tables in the database.

The **-pT** option prints the same information as the **-pt** option. In addition, the **-pT** option displays:

- Index-specific information
 - Page-allocation information by page type (for dbspaces)
 - The number of any compressed rows in a table or table fragment and the percentage of table or table-fragment rows that are compressed
- If table or fragment rows are not compressed, the "Compressed Data Summary" section does not appear in the output.

Plan when you want to run the **-pT** option, because it forces a complete scan of partitions.

Output for both **-pt** and **-pT** contains listings for **Number of pages used**. The value shown in the output for this field is never decremented because the disk space allocated to a tblspace as part of an extent remains dedicated to that extent even after you free space by deleting rows. For an accurate count of the number of pages currently used, refer to the detailed information on tblspace use (organized by page type) that the **-pT** option provides.

Example of oncheck -pt Output

The following example shows output of the **oncheck -pt** command:

TBLspace Report for testdb:tab1

Physical Address	2:10		
Creation date	10/07/2004	17:01:16	
TBLspace Flags	801	Page Locking	
		TBLspace use 4 bit bit-maps	
Maximum row size	14		
Number of special columns	0		
Number of keys	0		
Number of extents	1		
Current serial value	1		
Pagesize (k)	4		
First extent size	4		
Next extent size	4		
Number of pages allocated	340		
Number of pages used	337		
Number of data pages	336		
Number of rows	75806		
Partition partnum	2097154		
Partition lockid	2097154		
Extents			
Logical Page	Physical Page	Size	Physical Pages
0	2:106	340	680

Example of oncheck -pT Output

The following example shows output of the **oncheck -pT** command:

TBLspace Report for database_a:nilesh.table_1a

Table fragment partition dbspace1 in DBspace dbspace1

Physical Address	3:5		
Creation date	03/21/2009	15:35:47	
TBLspace Flags	8000901	Page Locking	
		TBLspace contains VARCHARS	
		TBLspace use 4 bit bit-maps	
		TBLspace is compressed	
Maximum row size	80		
Number of special columns	1		
Number of keys	0		
Number of extents	1		
Current serial value	100001		
Current SERIAL8 value	1		
Current BIGSERIAL value	1		
Current REFID value	1		
Pagesize (k)	2		
First extent size	8		
Next extent size	8		
Number of pages allocated	24		
Number of pages used	22		
Number of data pages	14		
Number of rows	500		
Partition partnum	3145730		

```

Partition lockid          3145730

Extents
  Logical Page      Physical Page      Size Physical Pages
           0          3:16053          24          24

Type          Pages      Empty Semi-Full      Full Very-Full
-----
Free          9
Bit-Map       1
Index         0
Data (Home)   14
Data (Remainder) 0          0          0          0          0
-----
Total Pages      24

Unused Space Summary

  Unused data bytes in Home pages      1177
  Unused data bytes in Remainder pages    0

Home Data Page Version Summary

      Version                      Count
      0 (current)                  14

Compressed Data Summary

Number of compressed rows and percentage of compressed rows  500 100.00

```

For more examples of using **oncheck -pt** and **-pT**, see managing disk space in the *IBM Informix Administrator's Guide* and the *IBM Informix Performance Guide*.

Related reference:

"check partition argument: Check partition consistency (SQL administration API)" on page 21-24

"print partition argument: Print partition information (SQL administration API)" on page 21-97

Turn On Locking with -x

The **-x** option can be appended to the **-ci**, **-cl**, **-pk**, **-pK**, **-pl**, and **-pL** options to place a shared lock on affected tables. While the table is locked, no other users can perform inserts, updates, and deletions while **oncheck** checks or prints the index. Without the **-x** option for tables with row locking, **oncheck** only places an IS (intent shared) lock on the table, which prevents actions such as dropping the table or the indexes during the check.

For example, the following sample command instructs **oncheck** to lock indexes for the **customer** table while it validates the order of key values, validates horizontal links, and ensures that no node appears twice in the index:

```
oncheck -cix stores_demo:customer
```

When you specify option **-x**, **oncheck** locks indexes for tables that use row locking. If **oncheck** detects page-lock mode, it displays a warning message and places a shared lock on the table regardless.

Send Special Arguments to the Access Method with -u

You can use the **-u** option to send special arguments to the access method. The possible arguments depend on the access method. For example, the R-tree access method supports the **display** option, as the following example shows:

```
oncheck -pl -u "display"
```

Use commas to separate multiple arguments in the argument string.

For information on valid arguments for your access method, refer to the user manual for your access method.

Return Codes on Exit

The **oncheck** utility returns the following codes on exit.

```
GLS failures:-1  
Invalid serial/key:2  
Onconfig access error:2  
Invalid onconfig settings:2  
Invalid arguments to oncheck:2  
Error connecting database server:1  
error detected by oncheck:2  
no errors detected by oncheck:0
```

Windows only:

```
Not properly installed:1  
Authentication error:2
```

Chapter 9. The onclean Utility

Use the **onclean** utility to force a shut down of the database server when normal shut down with the **onmode** utility fails or when you cannot restart the server. The **onclean** utility attempts to clean up shared memory, semaphores, and stops database server virtual processes.

Syntax

On UNIX and Linux, you must be user **root** or **informix** to run the **onclean** command. On Windows, you must be in the **Informix-Admin** group to run the command.



Table 9-1. Syntax Elements of the **onclean** Command

Element	Purpose
-k	Shuts down a server that is online by stopping database server virtual processes and attempting to clean up the remaining semaphores and shared-memory segments, even if they are still running.
-V	Displays short version information.
-version	Displays full version information.
-y	Does not prompt for confirmation.

Usage

Use the **onclean** utility to stop the database server only if the **onmode** utility is unable to shut it down or you cannot restart the server. Perhaps the database server shut down in an uncontrolled way and cannot recover, or it is hung. If the database server fails to restart, the previous instance of the database server is still attached to the shared-memory segments. Check the message log to see if the database server shut down abnormally. The **onclean** utility stops all **oninit** processes and attempts to remove all shared-memory segments and semaphores that are recorded in the **\$INFORMIXDIR/etc/conf.\$INFORMIXSERVER** file.

Attention: Use the **onclean** utility with caution. When you run **onclean**, any pending transactions and processes fail to complete, and user sessions are disconnected abruptly. However, the database server rolls back transactions when it restarts.

The **INFORMIXDIR**, **INFORMIXSERVER**, **INFORMIXSQLHOSTS**, and **ONCONFIG** environment variables must be set with valid values to run this utility.

The **onclean** command you should use depends on the situation:

- If you are not sure whether the database server is offline, use the **onclean** command without options. If the database server is still online, a message appears directing you to run the **onclean -k** command.
- If the database server is offline, use the **onclean** command.
- If the database server is online and you are sure that you want to force it to shut down, use the **onclean -k** command.

You can only use the **onclean** utility to shut down the local database server; you cannot use it to shut down a remote database server. The **onclean** utility should not be used to shut down an entire high-availability cluster or a remote database server.

The **onclean** utility might not be able to clean up shared memory segments that were in use by the database server in every situation. The **onclean** utility attempts to terminate only **oninit** processes. The **onclean** utility does not succeed in the following situations:

- If a non-database server process is attached to the shared memory segment prior to running the **onclean** command, the **onclean** utility does not stop this process to remove the shared memory segment.
- The **onclean** might not be able to guarantee a clean server startup is when an application or database server utility is connected to a network port. If the user tries to initialize a database server instance on the same network port, then the database server cannot start the listener thread and fails to start. The **onclean** utility does not stop the application to free the network port.

You can automate shutting down the database server with the **onshutdown** script, which calls the **onclean -ky** command if necessary.

Return Codes

- | | |
|---|--|
| 0 | Successful |
| 1 | Failure because of one of the following problems: <ul style="list-style-type: none"> • Incorrect environment variable settings • Incorrect privileges to run the onclean command • Incorrect command syntax • Corrupted information • Running the onclean command without the -k option on a server that is still online |
| 2 | Failure because one or more OS system calls used by onclean returned an error. |

Related reference:

“Taking the Database Server to Offline Mode with the **-k** Option” on page 14-16

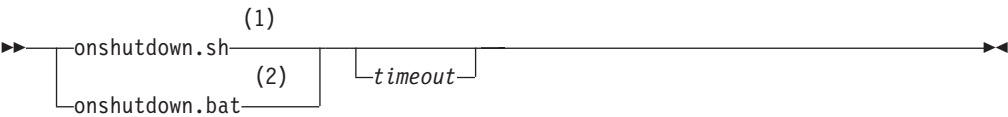
The onshutdown script

Use the **onshutdown** script to automate shutting down the database server. The script attempts to shut down the server normally. If the server has not shut down after a specified time, the script forces the server to shut down.

Syntax

The **onshutdown** script first runs the **onmode -ky** command. After a specified wait time, the script runs the **onclean -ky** command.

On UNIX and Linux, you must be user **root** or **informix** to run the **onshutdown** script. On Windows, you must be in the **Informix-Admin** group to run the **onshutdown** script.



Notes:

- 1 UNIX
- 2 Windows

Table 9-2. Syntax Elements of the **onshutdown** Script

Element	Purpose
<i>timeout</i>	The number of seconds after the onmode -ky command has been run before running the onclean -ky command. Must be a positive integer from 10 to 60. The default value is 30 seconds.

Usage

Use the **onshutdown** script only when forcing the database server to shut down would be appropriate.

Attention: Use the **onshutdown** script with caution. If the script needs to run the **onclean -ky** command, any pending transactions and processes fail to complete, and user sessions are disconnected abruptly. However, the database server rolls back transactions when it restarts.

The **INFORMIXDIR**, **INFORMIXSERVER**, **INFORMIXSQLHOSTS**, and **ONCONFIG** environment variables must be set with valid values to run this utility.

You can only use the **onshutdown** script to shut down the local database server; you cannot use it to shut down a remote database server. The **onshutdown** script should not be used to shut down an entire high-availability cluster or a remote database server instance.

The **onshutdown** script has a 10 second time period during which it can be aborted.

Related reference:

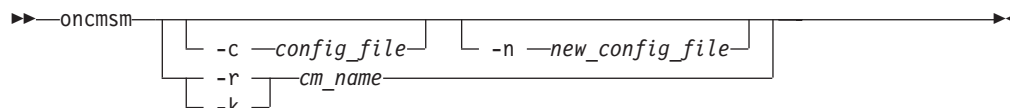
“Taking the Database Server to Offline Mode with the -k Option” on page 14-16

Chapter 10. The oncmsm utility

Use the **oncmstm** utility to start the Connection Manager. You configure the Connection Manager by using a configuration file.

Syntax

UNIX syntax diagram:



Windows syntax diagram:



Element	Purpose	Key Considerations
-c <i>config_file</i>	Identifies the path and file name of a configuration file that contains service level agreements and failover options.	If the configuration file name is omitted, the Connection Manager attempts to load the file from the following path and file name: \$INFORMIXDIR/etc/cmsm.cfg
<i>cm_name</i>	Specifies the name of the Connection Manager instance.	
-i	Installs the Connection Manager as a Windows service.	This option is valid for Windows platforms only.
-k <i>cm_name</i>	Shuts down a specific instance of the Connection Manager.	Use <i>cm_name</i> to specify the name of the Connection Manager instance.
-n <i>new_config_file</i>	Specifies the path and file name of a configuration file that will contain a converted version of an old Connection Manager configuration file.	Use this option to convert configuration files to the latest Connection Manager format.
-r <i>cm_name</i>	Reloads the Connection Manager settings.	Use the -r option to reload Connection Manager settings without stopping and restarting the Connection Manager.
-u	Uninstall the Connection Manager Windows service.	This option is valid for Windows platforms only.

Usage

Run the **oncmsm** utility from the command line to initialize the Connection Manager. For more information and examples of using the **oncmsm** utility, refer to the *IBM Informix Administrator's Guide*.

You can add, change, or delete Service Level Agreements (SLAs) while the Connection Manager is running by modifying the Connection Manager configuration file and reload it by using the **-r** option. You can reuse an SLA name after it has been deleted.

UNIX Only: Only user **informix** can run the **oncmsm** command. If user **root** or a member of the DBSA group is granted privileges to connect to the sysadmin database, then user **root** or that member of the DBSA group can also run **oncmsm**.

Windows Only:

- You must be a member of the **Informix-Admin** group to run the **oncmsm** utility. If user **administrator** or a member of the DBSA group is granted privileges to connect to the sysadmin database, then user **administrator** or that member of the DBSA group can also run **oncmsm**.
- You must first install **oncmsm** as a service. Two additional options can be passed on the **oncmsm** command line. Use the **-i** option to install **oncmsm** as a Windows service, and **-u** to uninstall the service. After **oncmsm** is installed as a Windows service, it can be started from either a command line by running **oncmsm** or it can be started from the **Services** program (click **Start**, click **Control Panel**, double-click **Administrative Tools**, and then double-click **Services**).

For increased readability of the configuration file, you can break long lines into smaller ones by using a backslash (\) as a line continuation character. The following example shows a macro definition separated into four lines:

```
MACRO  srv1list=node1,node2,node3,node4, \
        node6,node7,node8,           \
        node100,node101,              \
        node500,node501
```

Connection Manager configuration file

Most of the Connection Manager options are specified in a configuration file. The configuration file consists of two main parts: the heading and the body. The heading contains Connection Manager configuration options that specify the instance name and other optional parameters. The body consists of one or more connection unit sections that define the type and name of the connection unit, service level agreements, failover configuration and so on. The general format of the configuration file is as follows:

```
# *** HEADING ***
NAME                                cm_name

# Optional configuration file parameters
MACRO                               macro_name=(macro_list)
LOG                                 log_errors
LOGFILE                             log_file_path_and_file_name
DEBUG                               debug_value
EVENT_TIMEOUT                       timeout_value

# *** BODY ***
# Connection Unit 1
Group_type                          connection_unit_name
```

```

{
  INFORMIXSERVER  server_list
  SLA              SLA_name SLA_attribute=SLA_value
  FOC              ORDER=server_list | DISABLED | NOTIFY
                  TIMEOUT=FOC_timeout
                  RETRY=FOC_retry
  CMALARMPROGRAM  event_alarm_program
}
# Connection Unit 2
Group_type        connection_unit_name
{
  INFORMIXSERVER  server_list
  SLA              SLA_name SLA_attribute=SLA_value
  FOC              ORDER=server_list | DISABLED | NOTIFY
                  TIMEOUT=FOC_timeout
                  RETRY=FOC_retry
  CMALARMPROGRAM  event_alarm_program
}
.
.
.
# Connection Unit n
Group_type        connection_unit_name
{
  INFORMIXSERVER  server_list
  SLA              SLA_name SLA_attribute=SLA_value
  FOC              ORDER=server_list | DISABLED | NOTIFY
                  TIMEOUT=FOC_timeout
                  RETRY=FOC_retry
  CMALARMPROGRAM  event_alarm_program
}

```

Connection manager configuration file parameters

The table lists the parameters that you can use in different sections of the Connection Manager configuration file. For more information about each parameter, follow the link to it from the table. The parameters are described in alphabetical order after the table.

Table 10-1. Mandatory and optional parameters for different parts of the Connection Manager configuration file

Part of the configuration file	Mandatory parameters	Optional parameters
Heading configuration file parameters	NAME	LOGFILE DEBUG LOG MACRO TYPE CM_TIMEOUT SQLHOSTS SLA_WORKERS EVENT_TIMEOUT SECONDARY_EVENT_TIMEOUT CMALARMPROGRAM

Table 10-1. Mandatory and optional parameters for different parts of the Connection Manager configuration file (continued)

Part of the configuration file	Mandatory parameters	Optional parameters
Section for a high-availability cluster	SLA	EVENT_TIMEOUT
	FOC	CMALARMPROGRAM
	INFORMIXSERVER	SLA_WORKERS
Section for a grid, replicate set, or server set	SLA	EVENT_TIMEOUT
	INFORMIXSERVER	CMALARMPROGRAM
		SLA_WORKERS

CMALARMPROGRAM=*event_alarm_program*

Specifies the path and file name of a program or script to run if failover processing encounters an error. If failover processing fails, then the Connection Manager calls the program specified by CMALARMPROGRAM.

CM_TIMEOUT=*value*

Specifies the amount of time, in seconds, for the primary server to wait for events from the Connection Manager. If the primary server does not receive any events from the Connection Manager within the specified period of time, the server assumes that the Connection Manager is inoperative. If the inoperative Connection Manager is the failover arbitrator, then the next available Connection Manager becomes the failover arbitrator. If CM_TIMEOUT is not specified, the primary server waits 60 seconds before beginning failover processing.

DEBUG *value*

Specifies whether to enable or disable debug mode. Set the **DEBUG** parameter to 1 to log SQL and ESQL/C error messages. Set the **DEBUG** parameter to 0 or remove the **DEBUG** parameter from the configuration file to disable debug mode. You cannot enable debug mode from the command line.

EVENT_TIMEOUT=*timeout_value*

Specifies the amount of time, in seconds, for the Connection Manager to wait for events from the primary server before beginning failover processing. An *event* in this context is an indication from the primary server that the server is still functioning, such as a performance statistics message, a node change message, or an administration message. If the Connection Manager does not receive any events from the primary server within the specified period of time, the Connection Manager assumes that the primary server is inoperative and failover processing starts. If the **EVENT_TIMEOUT** parameter is not specified, the Connection Manager waits 60 seconds before beginning failover processing. If the **EVENT_TIMEOUT** parameter is set to -1, the Connection Manager waits indefinitely for events from the primary server, or until it receives notification from one of the secondary servers that the primary server is off line. The minimum value for the **EVENT_TIMEOUT** parameter is 30 seconds. The Connection Manager sets the **EVENT_TIMEOUT** parameter to 30 seconds if a value less than 30 is specified.

FOC ORDER=*server_list* | **DISABLED** | **NOTIFY**
TIMEOUT=*FOC_timeout*

RETRY=FOC_retry

Specifies the failover configuration (FOC) that the Connection Manager uses to determine which server should take over in case the primary server fails.

Important: If this parameter is enabled and failover processing occurs, you must not restart the failed primary server manually until failover processing completes.

The FOC parameter must be set to DISABLED if the connection unit type is REPLSET.

If the FOC parameter is not specified, the following default value is used:

FOC ORDER=SDS,HDR,RSS TIMEOUT=0 RETRY=1

Table 10-2. FOC parameters

Parameter	Description
ORDER=	<p>SDS Specifies an SD (Shared Disk) secondary server</p> <p>HDR HDR (High-Availability Data Replication) secondary server</p> <p>RSS RS (Remote Standalone) secondary server</p> <p>DISABLED Specifies that failover processing by the Connection Manager is disabled. If failover is disabled, the Connection Manager ignores failover requests.</p> <p>NOTIFY Specifies the Connection Manager to send alarm messages to the parameter specified by the CMCMALARMPROGRAM parameter.</p> <p>The ORDER parameter determines the order in which secondary servers take over the role of the primary server in case the primary server fails. The ORDER parameter is specified using a comma-separated list of servers or server alias names, a server type, or a previously defined macro.</p> <p>For example, the following example indicates that the HDR server should take over if the primary server fails, followed by the first available shared disk secondary server (SDS), followed by the list of servers defined by the CA macro, followed by the server known as srv1:</p> <p>FOC ORDER=HDR,SDS,{CA},srv1</p>
TIMEOUT=	<p><i>value</i> Specifies the number of seconds to wait before initiating the failover process.</p> <p>The TIMEOUT parameter applies only after the value of EVENT_TIMEOUT is exceeded. For example, assume that the EVENT_TIMEOUT parameter is set to 60 and the failover timeout value is set to 10. If the Connection Manager does not receive any events from the primary server for 60 seconds, another 10 seconds must elapse before failover processing begins. See “Failover Configuration Examples” on page 10-15 for more information.</p>

Table 10-2. FOC parameters (continued)

Parameter	Description
RETRY=	<i>value</i> Specifies the number of times to go through the ORDER list while waiting for a successful failover to complete. Setting a value of 0 indicates that the Connection Manager should continue retrying failover processing indefinitely: After the RETRY limit has been reached without a successful failover, an alarm is generated and the Connection Manager terminates the automated failover attempt.

INFORMIXSERVER *server_list*

Specifies an Informix server or a list of servers to connect with during initialization of the Connection Manager. If the first server in the list is inoperative, then the Connection Manager attempts to connect to the next server in the list, and so on until the end of the list is reached. You can use a macro for the server list of a comma-separated server list as in the following:

```
INFORMIXSERVER ${NYservers}
```

Group_type *connection_unit_name*

A connection unit is a specific high-availability server configuration. You specify connection units in your Connection Manager configuration file to direct client connection requests to the appropriate database server. Specify the *Group_type* as one of the following:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

The *connection_unit_name* specifies the name by which the connection unit is known to the Connection Manager. Each *connection_unit_name* name must be unique within the configuration file.

When specifying the name of a CLUSTER connection unit in the configuration file, you should use the same name as the group name for the corresponding cluster in the INFORMIXSQLHOSTS file.

When specifying connection unit names in the configuration file, you can use multibyte characters only for GRID and REPLSET connection unit types.

LOG *log_errors*

Specifies whether to log information about PROXY and REDIRECT modes.

- LOG=1 logs information about both PROXY and REDIRECT service level agreements.
- LOG=2 or LOG=3 logs PROXY mode information only.
- LOG=2 logs data communication between the client and the server.
- LOG=3 logs data communication between the client and the server, and logs the data buffer.

LOGFILE *log_file_path_and_file_name*

Specifies the name and location of the Connection Manager log file. The log file is continuously updated with status information as long as the

Connection Manager is running. The log file is necessary for monitoring and troubleshooting purposes. Make sure that the directory for the log file exists, and that access to the file is enabled for the user starting the Connection Manager. If no log file is specified, the Connection Manager creates a new file based on the Connection Manager name and the process ID. The path and file name of the log file are displayed when the Connection Manager is started.

MACRO *macro_name*=(*macro_list*)

Specifies the name of a macro and a list of server names. The macro name expands to the list of server names when used elsewhere in the configuration file. For example, the following line defines a macro named CA that consists of three servers:

```
MACRO CA=(ca1,ca2,ca3)
```

To use the macro, enclose the macro name in braces as shown here:

```
SLA rep11_ca DBSERVERS=${CA}
```

NAME *cm_name* (**Mandatory**)

Specifies the name of the Connection Manager instance. If the NAME parameter is not specified, the name of the first SLA is used as the Connection Manager instance name. You use *cm_name* to monitor the instance and to shut down or reload the Connection Manager. If more than one Connection Manager instance is configured, the name of each instance must be unique.

SECONDARY_EVENT_TIMEOUT=*timeout_value*

Specifies the amount of time, in seconds, that the Connection Manager waits for events from a secondary server before assuming the server has failed. If the Connection Manager receives no events from a secondary server during the time specified, the Connection Manager disconnect from the secondary server.

SLA *SLA_name* *SLA_attribute*=*SLA_value*

Specifies the service level agreement (SLA) name, redirection policy, and redirection type. Use the *SLA_name* parameter to specify the name of the SLA. The redirection policy specifies the type or the name of the target database server to which the client connects. Client applications use the SLA name to connect to the appropriate database server. The SLA parameter can be specified multiple times in the same configuration file; however, each SLA name must be unique. See “Service Level Agreement Examples” on page 10-11.

Each SLA must have a corresponding entry in the **sqlhosts** file. For each SLA, a listener thread is installed at the specified port on the server to detect incoming client requests. The Connection Manager connects clients to the appropriate database server according to the redirection policy.

Table 10-3. SLA parameters

Attribute	Value
DBSERVERS=	<p>SDS Specifies an SD (Shared Disk) secondary server type.</p> <p>HDR Specifies an HDR (High-Availability Data Replication) secondary server type.</p> <p>RSS Specifies an RS (Remote Standalone) secondary server type.</p> <p>ANY Specifies any available Informix database server.</p> <p>PRI or primary Specifies the primary server.</p> <p><i>server_name</i> or <i>server_alias</i> Specifies the name or alias name of a specific server.</p> <p>\${macro}</p> <p>The DBSERVERS parameter is specified using a comma-separated list of servers or server alias names, a server type, or a previously defined macro.</p>
HOST=	<p><i>hostname</i> Specifies a particular host or host alias name.</p> <p><i>TCP/IP_address</i> Specifies a TCP/IP address.</p>
MODE=	<p>Specifies how Connection Manager redirects client connection requests.</p> <p>PROXY Specifies that client connections are to use proxy mode when clients connect to Informix database servers from outside a firewall. Setting proxy mode causes the Connection Manager to manage client communication with the appropriate server. Use proxy mode when the server is situated behind a firewall that the client would not ordinarily be able to connect to. Because the Connection Manager handles all client requests, consider configuring multiple Connection Manager instances when using proxy mode.</p> <p>REDIRECT Specifies that client connections to use redirect mode, which configures the Connection Manager to return the appropriate server node, IP address, and port number to the client application (depending on the SLA definition). The client application connects to the server using the IP address and port number that the Connection Manager provides.</p>

Table 10-3. SLA parameters (continued)

Attribute	Value
NETTYPE=	<p>onsoctcp Specify sockets with TCP/IP protocol.</p> <p>drsotcp Specifies Distributed Relational Database Architecture (DRDA) - connection for IBM data server client.</p> <p>onsocssl Specifies Secured Sockets Layer (SSL) protocol for DRDA.</p> <p>By default, the Connection Manager reads network configuration data from the \$INFORMIXSQLHOSTS file; however, you can specify network information in the Connection Manager configuration file without having to modify the \$INFORMIXSQLHOSTS file.</p>
POLICY=	<p>LATENCY Specifies that the Connection Manager will direct connection requests to the server with the lowest latency.</p> <p>FAILURE Specifies that connection requests will be routed to the server with the fewest apply failures.</p> <p>WORKLOAD Specifies that connection requests will be directed to the server with the lowest workload.</p> <p>The POLICY parameter specifies how the Connection Manager prioritizes client connection requests. The policy function uses a formula to decide which server to direct clients; the server with the best value is used. The policy refers to a relative value rather than an absolute value. For example, specifying LATENCY does not indicate a specific latency period.</p> <p>To select the server with the lowest latency and the fewest apply failures, making the LATENCY option 10 times more important than the FAILURE option, use: POLICY=10*LATENCY+FAILURE</p> <p>To use the LATENCY and FAILURE policies, you must have created a table through the grid with replication enabled.</p> <p>Quality of data monitoring must be enabled to use the POLICY parameter. See Routing client connections in a grid (Enterprise Replication).</p>
SERVICE=	<p><i>port_number</i> Specifies a port number</p> <p><i>servicename</i> Specifies a service name</p>

Table 10-3. SLA parameters (continued)

Attribute	Value
SQLHOSTSOPT=	<p>Specifies optional connectivity information. Enclose optional parameters in quotation marks.</p> <p>For example, suppose you wanted the following SQLHOSTS options configured in your SLA:</p> <pre># server nettype hostname servicename options informix1 onsoctcp host_1 sales_1 csm=cmspath</pre> <p>You can specify the same parameters in your SLA using the following SLA definition:</p> <pre>SLA oltp1 DBSERVERS=informix1 \ NETTYPE=onsoctcp \ HOSTNAME=host_1 \ SERVICE=sales_1 \ SQLHOSTSOPT="csm=cmspath"</pre> <p>See sqlhosts Connectivity information (Administrator's Guide) for more information.</p>
WORKERS=	<p><i>value</i> Specifies the number of threads allocated to the SLA. When a service level agreement is specified, the Connection Manager creates an SLA listener process to intercept client connection requests. The SLA listener process can have one or more threads, called worker threads. If WORKERS is not specified, the number of worker threads assigned is 4. To set the number of worker threads to 8, set the following:</p> <pre>WORKERS=8</pre>

SLA_WORKERS

Specifies the number of worker threads. When a service level agreement is specified, the Connection Manager creates an SLA listener process to intercept client connection requests. The SLA listener process can have one or more threads, called worker threads. If the **SLA_WORKERS** parameter is not specified, the number of worker threads assigned is 8.

SQLHOSTS *option*

Specify **SQLHOSTS local**, **SQLHOSTS remote**, or **SQLHOSTS local+remote**. When the Connection Manager receives a client connection request, it searches for the requested server based on the list of server instances defined in the sqlhosts file located on the Connection Manager instance. You can use the SQLHOSTS parameter to cause the Connection Manager to search for database instances on servers other than those listed in the sqlhosts file on the Connection Manager server instance.

For example, assume you have a web server located on an external network and an application server located on an internal network. The database server defines TCP/IP port1 and port2 for external network connections and TCP/IP port3 and port4 for internal connections. Specifying **SQLHOSTS local** causes the Connection Manager to find server instances defined only in the sqlhosts file of the Connection Manager instance. In this case, the Connection Manager directs client connection requests only to servers on the external network.

Specifying **SQLHOSTS remote** causes the Connection Manager to find server instances defined in the sqlhosts file of the primary server instance. Specifying **SQLHOSTS local+remote** causes the Connection Manager to direct connection requests to servers defined in the sqlhosts files of both the Connection Manager instance and the primary server instance.

The SQLHOSTS option is useful when you want to prevent users from accessing one or more servers in the cluster.

Table 10-4. Values for the SQLHOSTS Connection Manager configuration parameter

SQLHOSTS parameter	Result
local	The Connection Manager searches for the requested instance in the sqlhosts file located on the computer on which the Connection Manager is running. The Connection Manager uses the value set by the INFORMIXSQLHOSTS environment variable to determine which sqlhosts file to use; Informix instances are specified by the DBSERVERNAME or DBSERVERALIASES configuration parameters.
remote	The Connection Manager searches for the requested instance in the sqlhosts file located on a remote instance. The Connection Manager uses the value set by the INFORMIXSQLHOSTS environment variable to determine which sqlhosts file to use. Informix instances are specified by the DBSERVERNAME or DBSERVERALIASES configuration parameters.
local+remote	The Connection Manager first checks the local sqlhosts file for Informix instances; if the requested server entry is not found then the DBSERVERNAME or DBSERVERALIASES configuration parameters on the remote server are used.

Service Level Agreement Examples

The examples in this section show how to configure SLAs for high-availability clusters, replicate sets, grids, and server sets.

High-availability cluster examples

In each of the following examples, it is possible to use specific server alias names or generic server types. For example, specifying the keyword **RSS** indicates a generic server type, which means that the Connection Manager evaluates every RS secondary server in the cluster to determine the best candidate for client applications requests. You can also refer to a specific server by specifying its alias, such as *sds1*, or *rss1*.

The following examples show how the configuration file should appear. If the configuration file was named `cm_config`, you would start the Connection Manager by typing the following at a command prompt:

```
oncmsm -c cm_config
```

Example 1:

This example configures the Connection Manager for a high-availability cluster and defines two service level agreements. The heading of the configuration file defines the name, `cm1`, specifies that logging is to occur, and specifies the path and file name of the log file. The body of the configuration file defines two SLAs. The first SLA directs client connection requests such as connect to `@oltp` to the primary

server. The second SLA directs connection requests such as connect to @report to the HDR secondary server or, if the HDR secondary server is not available, to any available SD secondary server.

```
NAME      cm1
LOG        1
LOGFILE    ${INFORMIXDIR}/tmp/cm1.log

CLUSTER west # cluster name
{
  INFORMIXSERVER ids_w1,ids_w2
  SLA oltp DBSERVERS=primary
  SLA report DBSERVERS=(HDR,SDS)
  FOC ORDER=SDS,HDR TIMEOUT=5 RETRY=1
  CMALARMPROGRAM ${INFORMIXDIR}/etc/CMALARMPROGRAM.sh
}
```

Example 2:

This example configures the Connection Manager to resolve a client request such as: connect to @secondaryNodes by connecting to one of the SD secondary servers. If no SD secondary servers exist, then the Connection Manager attempts to direct the connection to the HDR secondary server. If the HDR secondary server is inoperative, then the Connection Manager attempts to connect to the primary server.

```
NAME      cm1
LOG        1
LOGFILE    ${INFORMIXDIR}/tmp/cm1.log

CLUSTER west # cluster name
{
  INFORMIXSERVER ids_w1,ids_w2
  SLA secondaryNodes DBSERVERS=SDS,HDR,primary
  FOC ORDER=SDS,HDR TIMEOUT=5 RETRY=1
  CMALARMPROGRAM ${INFORMIXDIR}/etc/CMALARMPROGRAM.sh
}
```

Example 3:

To enable load balancing among servers, use parentheses around server names or server types. In this example, the parentheses around SDS,HDR indicates that the connection could be routed to an SD secondary server or to the HDR secondary server based on which server has the lowest CPU utilization.

```
NAME      cm1
LOG        1
LOGFILE    ${INFORMIXDIR}/tmp/cm1.log

CLUSTER west # cluster name
{
  INFORMIXSERVER ids_w1,ids_w2
  SLA secondary DBSERVERS=(SDS,HDR)
  FOC ORDER=SDS,HDR TIMEOUT=5 RETRY=1
  CMALARMPROGRAM ${INFORMIXDIR}/etc/CMALARMPROGRAM.sh
}
```

Example 4:

In this example, assume that the primary server is assigned an alias name of **alpha**, the HDR secondary server has an alias of **beta**, and there are two SD secondary servers with aliases of **gamma1** and **gamma2**. The Connection Manager is started with three service level agreements, **sla1**, **sla2**, and **sla3**. Client connection requests sent to **sla1** are directed to the primary server using REDIRECT mode, meaning

that the Connection Manager sends the IP address and port number of the primary server to the client application, which uses the information to connect to the server. Client connection requests sent to **sla2** are sent to the HDR secondary server (beta) using PROXY mode, which means that the Connection Manager is responsible for routing client requests. Connection requests sent to **sla3** are sent to one of the SD secondary servers (gamma1 or gamma2) based on which server has the most available resources.

```
NAME      cm1
LOG        1
LOGFILE    ${INFORMIXDIR}/tmp/cm1.log

CLUSTER west # cluster name
{
  INFORMIXSERVER ids_w1,ids_w2
  SLA  sla1 DBSERVERS=alpha MODE=REDIRECT
  SLA  sla2 DBSERVERS=beta  MODE=PROXY
  SLA  sla3 DBSERVERS=SDS
  FOC  ORDER=SDS,HDR TIMEOUT=5 RETRY=1
  CMALARMPROGRAM ${INFORMIXDIR}/etc/CMALARMPROGRAM.sh
}
```

Start the Connection Manager from a command prompt using the following command:

```
oncmsm -c connectmgr
```

When the Connection Manager starts using the *connectmgr* configuration file, the following occurs:

- A Connection Manager instance named **cm1** is created.
- Four service level agreements, named **oltp**, **report**, **primary**, and **secondary** are created.
 - Client requests sent to SLA **oltp** are directed to the primary server using the Connection Manager as a proxy server.
 - Client requests sent to SLA **report** are directed to the first available SD secondary server; if no SD secondary servers are available, connections are sent to any available RS secondary server; if no RS secondary servers are available, then client requests are sent to the HDR secondary server.
 - Client requests sent to SLA **primary** are directed to the primary server.
 - Client requests sent to SLA **secondary** are directed to the secondary server that has the most resources available, regardless of whether the server is an SD secondary server, RS secondary server, or HDR secondary server.
- Client connection requests sent to **report**, **primary**, and **secondary** are directed to the appropriate server using REDIRECT mode. REDIRECT mode is the default if no mode is specified.
- Connection Manager creates a log file named /tmp/cm1.log.
- Debug mode is enabled.

Replicate set example

The following example shows the Connection Manager configuration file for an ER replicate set.

```
NAME      cm1
MACRO     NY=(ny1,ny2,ny3)
MACRO     CA=(ca1,ca2,ca3)
LOG        1
LOGFILE    ${INFORMIXDIR}/tmp/cm1.log
```

```

REPLSET erset #replset name
{
    INFORMIXSERVER    g_er1,g_er2
    SLA repl1_any      DBSERVERS=ANY # node discovery
    SLA repl1_ca       DBSERVERS=${CA} \
        POLICY=WORKLOAD
    SLA repl1_ny       DBSERVERS=${NY}
}

```

Grid example

This is an example configuration file for a grid.

```

NAME      cm1
MACRO     NY=(ny1,ny2,ny3)
MACRO     CA=(ca1,ca2,ca3)
LOG       1
LOGFILE   ${INFORMIXDIR}/tmp/cm1.log

GRID grid1 # grid name
{
    INFORMIXSERVER    node1,node2,node3
    SLA grid1_any      DBSERVERS=ANY POLICY=LATENCY
    SLA grid1_avail    DBSERVERS=${NY},${CA}
}

```

Server set example

This example shows an example configuration to support a server set. The Connection Manager provides availability and load balancing of the servers specified in the SLA. The server set must be supported by the Connection Manager.

```

NAME      cm1
MACRO     NY=(ny1,ny2,ny3)
MACRO     CA=(ca1,ca2,ca3)
LOG       1
LOGFILE   ${INFORMIXDIR}/tmp/cm1.log

SERVERSET ss # name of serverset, unique in CM
{
    INFORMIXSERVER ids1,ids2,ids3
    SLA ssavail DBSERVERS=ids1,ids2,ids3 \
        HOST=apollo SERVICE=9600 NETTYPE=onsoctcp
    SLA ssany DBSERVERS=(ids1,ids2,ids3) \
        HOST=apollo SERVICE=9610 \
        NETTYPE=onsoctcp
}

```

Multiple connection unit example

This example shows how the Connection Manager is used to manage multiple connection units, including multiple high-availability clusters, replicate sets, grids, and server sets. The example configures the Connection Manager to support two high-availability clusters, a replicate set, two grids, and one server set.

```

NAME      cm1
MACRO     NY=(ny1,ny2,ny3)
MACRO     CA=(ca1,ca2,ca3)
LOG       1
LOGFILE   ${INFORMIXDIR}/tmp/cm1.log

CLUSTER west # cluster name (unique in CM)
{
    INFORMIXSERVER ids_w1,ids_w2
}

```



```

SLA oltpw DBSERVERS=primary
SLA reportw DBSERVERS=(HDR,SDS)
FOC ORDER=SDS,HDR TIMEOUT=5 RETRY=1
CMALARMPROGRAM ${INFORMIXDIR}/etc/CMALARMPROGRAM.sh
}
CLUSTER east # cluster name in CM
{
  INFORMIXSERVER ids_e1,ids_e2
  SLA oltpw DBSERVERS=primary
  SLA reporte DBSERVERS=RSS,HDR
  FOC ORDER=HDR,RSS TIMEOUT=5 RETRY=1
  CMALARMPROGRAM ${INFORMIXDIR}/etc/CMALARMPROGRAM.sh
}
REPLSET erset #replset name
{
  INFORMIXSERVER g_er1,g_er2
  SLA repl1_any DBSERVERS=ANY # node discovery
  SLA repl1_ca DBSERVERS=${CA} \
    POLICY=WORKLOAD
  SLA repl1_ny DBSERVERS=${NY}
}
GRID grid1 # grid name
{
  INFORMIXSERVER node1,node2,node3
  SLA grid1_any DBSERVERS=ANY POLICY=LATENCY
  SLA grid1_avail DBSERVERS=${NY},${CA}
}
GRID grid2 # grid name
{
  INFORMIXSERVER node4,node5
  SLA grid2_any DBSERVERS=ANY POLICY=LATENCY
  SLA grid2_avail DBSERVERS=${CA},${NY}
}
SERVERSET ss # name of serverset, unique in CM
{
  INFORMIXSERVER ids1,ids2,ids3
  SLA ssavail DBSERVERS=ids1,ids2,ids3 \
    HOST=apollo SERVICE=9600 NETTYPE=onsoctcp
  SLA ssany DBSERVERS=(ids1,ids2,ids3) \
    HOST=apollo SERVICE=9610 \
    NETTYPE=onsoctcp
}

```

Failover Configuration Examples

These examples show how to configure failover by using Connection Manager configuration files.

The following examples show several ways to configure Connection Manager to promote a secondary server to the primary server in the event the original primary server encounters a problem. Each SLA should have a failover configuration defined.

Example 1:

In this example, if the Connection Manager detects that the primary server is offline, then failover processing starts immediately because the timeout value is set to zero. The Connection Manager first attempts to convert the best available SD secondary server to the primary server; if no SD secondary servers are online, it attempts to convert the HDR secondary into the primary server; if that is not possible, it attempts to convert the most suitable RS secondary server to the primary server.

```
FOC ORDER=SDS,HDR,RSS TIMEOUT=0
```

Example 2:

In this example, setting the timeout value to 10 causes the Connection Manager to wait 10 seconds for the primary server to come back on line before starting failover processing. If the primary server does not come back online within the specified time, the Connection Manager attempts to convert any SD secondary server, any HDR secondary server, or any RS secondary server to the primary server.

```
FOC ORDER=SDS,HDR,RSS TIMEOUT=10
```

Example 3:

To specify a group of specific servers that can become the primary server, use alias names. In this example, the Connection Manager immediately performs a failover because the timeout value is not specified and is assumed to be 0. The Connection Manager attempts to fail over to the server with alias **hdrs1**; if **hdrs1** is not available, then it attempts to fail over to **sds1** if possible, and so on, trying **rss1**, **sds2**, and finally **rss2**.

```
FOC ORDER=hdrs1,sds1,rss1,sds2,rss2
```

Example 4:

In this example, if the Connection Manager is not able to determine within 20 seconds that the current primary server is online, then failover processing is started. The Connection Manager first attempts to convert the server with alias **sds1** to the primary server if possible; if not, then it attempts to convert the server with alias **hdrs1** to the primary server if possible; if not then the best SD secondary server is converted to the primary. The **RETRY** parameter indicates that the Connection Manager will go through the **ORDER** list a maximum of two times while waiting for a successful failover to complete. If failover does not complete after going through the **ORDER** list two times, an alarm is generated and the Connection Manager terminates failover.

Note that this failover configuration will not fail over to RS secondary servers because none are specified.

```
FOC ORDER=sds1,hdrs1,SDS TIMEOUT=20 RETRY=2
```

Proxy Mode and Redirect Mode

Use the **MODE=REDIRECT** or **MODE=PROXY** after the SLA definition to specify how the Connection Manager redirects client connection requests.

Specifying **MODE=REDIRECT** configures the Connection Manager to return the appropriate server node, IP address, and port number to the client application (depending on the SLA definition). The client application connects to the server using the IP address and port number that the Connection Manager provides. See Figure 10-1 on page 10-17.

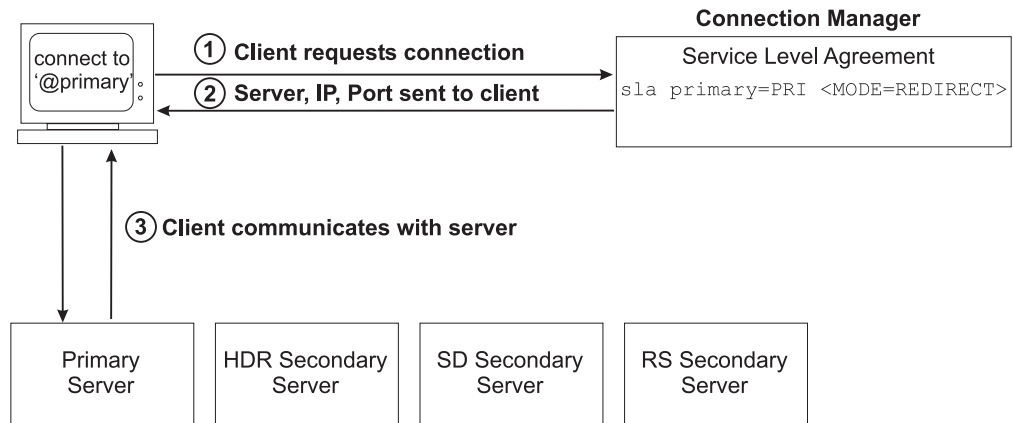


Figure 10-1. *MODE=REDIRECT* Service Level Agreement

Specifying **MODE=PROXY** configures the Connection Manager as a proxy server. Use **MODE=PROXY** when clients connect to Informix data servers from outside a firewall. Setting **MODE=PROXY** for a service level agreement causes the Connection Manager to manage client communication with the appropriate server. Use **MODE=PROXY** when the server is situated behind a firewall that the client would not ordinarily be able to connect to. Because the Connection Manager handles all client requests, consider configuring multiple Connection Manager instances when using **MODE=PROXY**. See Figure 10-2.

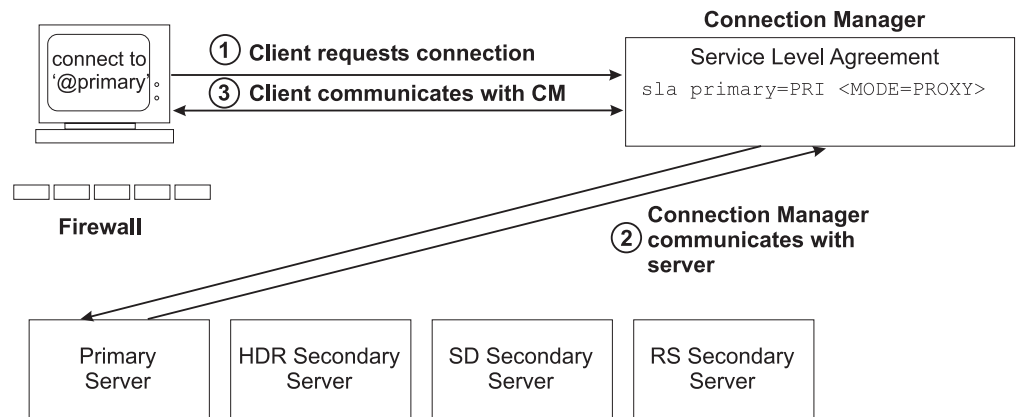


Figure 10-2. *MODE=PROXY* Service Level Agreement

Chapter 11. The ondblog Utility

In This Chapter

This chapter shows you how to use the ondblog utility.

ondblog: Change Logging Mode

Use the **ondblog** utility to change the logging mode for one or more databases.

Alternatively, you can change the logging mode by using an SQL administration API command with the **alter logmode** argument.

The **ondblog** utility logs its output in the BAR_ACT_LOG file.

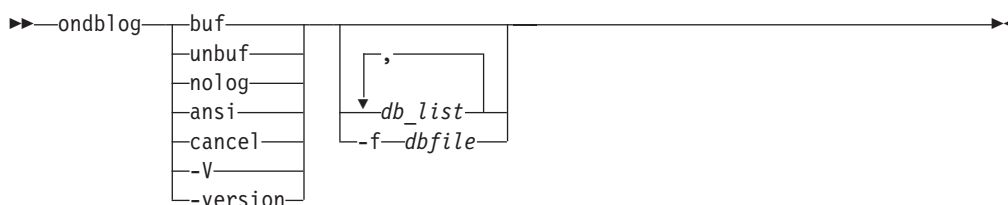
If you turn on transaction logging for a database, you must create a level-0 backup of all of the storage spaces that contain data in the database before the change takes effect.

For more information and examples of logging modes, see the following topics in the chapter on managing database-logging status in the *IBM Informix Administrator's Guide*:

- Modifying the database-logging status
- Modifying table-logging status

You cannot use the ondblog utility on High-Availability Data Replication (HDR) secondary servers, remote standalone (RS) secondary servers, or shared disk (SD) secondary servers.

ondblog Syntax



Element	Purpose	Key Considerations
buf	Sets the logging mode so that transaction information is written to a buffer before it is written to a logical log	None.
unbuf	Sets the logging mode so that data is not written to a buffer before it is written to a logical log	None.
nolog	Sets the logging mode so that no database transactions are logged	None.
ansi	Changes database logging to be ANSI compliant	Once you create or convert a database to ANSI mode, you cannot change it back to any of the other logging modes.

Element	Purpose	Key Considerations
cancel	Cancels the logging-mode change request before the next level-0 backup occurs	None.
-f <i>dbfile</i>	Changes the logging status of the databases that are listed (one per line) in the text file whose pathname is given by <i>dbfile</i>	This command is useful if the list of databases is long or used often.
<i>db_list</i>	Names a space-delimited list of databases whose logging status is to be changed	If you do not specify anything, all databases that the database server manages are modified.

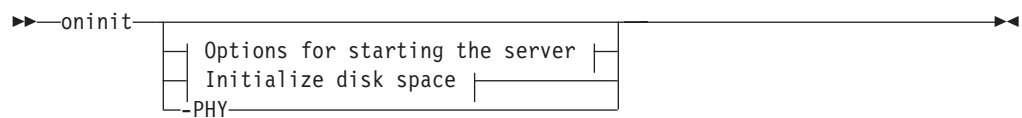
Chapter 12. The oninit utility

The **oninit** utility starts the database server.

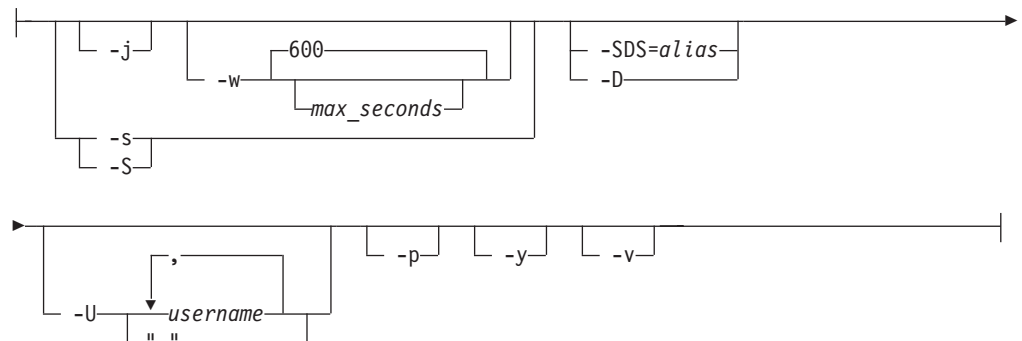
On UNIX, Linux, and Mac OS X, you must be logged in as user **root**, user **informix**, or the non-root database server owner to run the **oninit** utility. User **informix** should be the only member of the group **informix**. Run the **oninit** command from the command line.

On Windows, IBM Informix runs as a Windows service. Any user who has appropriate permissions to start a Windows service is able to start the IBM Informix service. The **Services** control application runs the **oninit** utility with any options that you supply.

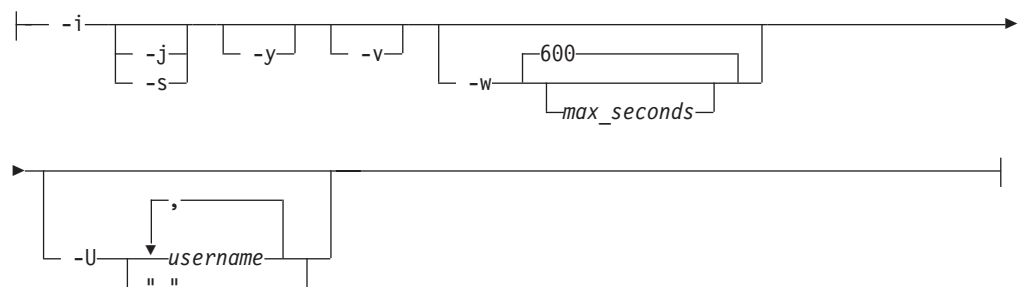
Syntax



Options for starting the server:



Initialize disk space:



Element	Purpose	Key Considerations
-D	Starts the database server with Enterprise Replication and high-availability cluster replication disabled.	
-i	Initializes disk space for the root dbspace so that it can be used by the database server and starts the database server.	<p>Disk space needs to be initialized only once to prepare data storage for the server.</p> <p>By default, to prevent data loss, you cannot reinitialize disk space. To reinitialize disk space for an existing root dbspace, you must set the FULL_DISK_INIT configuration parameter to 1 and then run the oninit -i command.</p> <p>See “Initialize disk space for the root dbspace” on page 12-3.</p>
-j	Starts the server in administration mode.	See “Start the server in administration mode ” on page 12-3.
-p	Starts the database server without deleting temporary tables.	If you use this option, the database server starts more rapidly, but space used by temporary tables left on disk is not reclaimed.
-PHY	Starts the server as of most current checkpoint. The -PHY option is used to tell the server to do only physical recovery without logical recovery.	<p>This option is normally used to start a secondary server. You must run one of the following commands to connect the secondary server to the primary server:</p> <pre>onmode -d secondary onmode -d RSS</pre> <p>The connection of the secondary server to the primary server fails if the most recent checkpoint on the primary server was not performed on the secondary server.</p>
-s	Starts the server in quiescent mode.	<p>The database server must be shut down when you use this option.</p> <p>When the database server is in quiescent mode, only the user informix can access the database server.</p>
-S	Starts database server in quiescent mode as a standard server with high-availability data replication disabled.	When the database server is in quiescent mode, only the user informix can access the database server.
-SDS=<i>alias</i>	For shared disk servers, starts the current server and specifies the primary server with the alias name.	When both the primary server and all of the SDS servers are down, use the -SDS=<i>alias</i> option to start the designated SDS server as the primary server. The -SDS=<i>alias</i> flag cannot be combined with the -i flag.
-U <i>username</i>	Specifies which users can access the server in administration mode for the current session.	<p>The informix user and members of the DBSA group are always administration mode users.</p> <p>See “Start the server in administration mode ” on page 12-3.</p>
-v	Displays verbose informational messages while the server is starting.	
-w <i>max_seconds</i>	Starts the database server and waits to indicate success or failure until the server is completely started in online mode or the number of seconds specified by <i>max_seconds</i> elapses.	<p>The default number of seconds to wait is 600.</p> <p>This option is not valid on secondary servers in a high-availability cluster.</p> <p>See “Start the server with a script” on page 12-4.</p>

Element	Purpose	Key Considerations
-y	Prevents verification prompts.	The -y option automatically answers yes to all the verification prompts.

Usage

By default, the **oninit** utility shows verification prompts during server startup. You can suppress verification prompts by including the **-y** option. You can view verbose informational messages by including the **-v** option. On UNIX, Linux, and Mac OS X, **oninit** output is shown to standard output. On Windows, you can view **oninit** output by setting the **ONINIT_STOUT** environment variable to save the output to a file.

You can start the server in different operating modes. By default, if you run the **oninit** command without options, the server starts in online mode. When the database server is in online mode, all authorized users can access the server.

Start the server in administration mode

Administration mode is an administrator-only mode you can use to perform maintenance operations including those that require running SQL or DDL commands. When in administration mode, the database server only accepts connection requests from the following users:

- The **informix** user
- Members of the DBSA group
- Users specified by the **oninit -U** command or the **onmode -j -U** command, for the current session. The **-U** option overrides any users listed by the **ADMIN_MODE_USERS** configuration parameter in the **onconfig** file.
- Users specified by the **ADMIN_MODE_USERS** configuration parameter

Use the **-U** option with a list of comma-separated user names to add administration mode users, such as: **Karin,Sarah,Andrew**.

Use the **-U ""** option to remove all administration mode users except the **informix** user and members of the DBSA group: **oninit -U ""**.

Initialize disk space for the root dbspace

The first time you install IBM Informix on your system, disk space for the root dbspace for the database server needs to be initialized. The root dbspace is specified by the **ROOTPATH** configuration parameter.

If you performed a typical installation and chose to create a database server or you performed a customer installation, disk space was automatically initialized. Otherwise, you must initialize disk space by running the **oninit -i** command.

If necessary, you can reinitialize disk space. Reinitializing disk space destroys all existing data managed by the database server. The database server must be offline when you reinitialize.

By default, you cannot reinitialize a root dbspace that is being used by the database server. Disk initialization fails if a page zero exists at the root path location (at the first page of the first chunk location). You can allow disk

reinitialization of an existing root dbspace by setting the FULL_DISK_INIT configuration parameter to 1.

Start the server with a script

You can use the **oninit -w** command in customized startup scripts and to automate startup. The **-w** option forces the server to wait until startup is completely successfully before indicating that the server is in online mode by returning to the shell prompt with a return code of 0. If the server is not in online mode within the timeout period, the server returns a return code of 1 to the shell prompt and writes a warning message in the online log.

The default timeout is 600 seconds (10 minutes), which you can modify to any integer value.

After running the following command, if the server fails to start within 60 seconds, a code of 1 is returned to the prompt:

```
oninit -w 60
```


To determine the reason for the server failing to start, check the online log. You might need to increase the timeout value. When you use the **oninit -w** command in a script, you can check whether the server is online with the **onstat - (Print output header)** command.

Related concepts:

 Database server operating modes (Administrator's Guide)

Related reference:

"FULL_DISK_INIT Configuration Parameter" on page 1-80

 Initialization process (Administrator's Guide)

Return codes for the oninit utility

If a **oninit** command encounters an error, the database server returns an error message and a return code value.

The following table contains the return codes, message text, and user actions for the **oninit** utility.

Table 12-1. Return codes for the oninit utility

Return Code	Message Text	User Action
0	The database server was initialized successfully.	The database server started.
1	Server initialized has failed. Look at any error messages written to stderr or the online message log.	Take the appropriate action based on the error messages written to stderr or the online message log.

Table 12-1. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
87	The database server has detected security violations or certain prerequisites are missing or incorrect.	(UNIX and Mac OS only) Check if user and group informix exists. Check if the server configuration file (onconfig) and sqlhosts file exists and has the correct permissions. Check if the environment variables INFORMIXDIR , ONCONFIG , and SQLHOSTS have a valid value and their length does not exceed 255 characters. Check if the environment variable INFORMIXDIR specifies an absolute path and does not have any spaces, tab, new lines, or other incorrect characters. Check if role separation-related subdirectories under the \$INFORMIXDIR directory, such as aaodir and dbssodir, have the correct ownership. Run the onsecurity utility to diagnose and fix any issues.
170	The database server failed to initialize the dataskip structure.	Free some physical memory on the system and try to start the database server again.
172	The database server failed to initialize the listener threads.	Free some system resources, check the configuration parameter values for the number of listener threads to start when the database server starts up, and try to start the database server again.
173	The database server failed to initialize data replication.	Free some physical memory in the system and try to start the database server again.
174	The database server failed to start fast recovery threads.	Free some physical memory in the system and try to start the database server again.
175	The database server failed to initialize the root dbspace.	Check the root dbspace related parameters in server configuration file (onconfig) to make sure that the path for the root dbspace is valid.
176	Shared disk secondary server initialization failed.	Check the entries in sqlhosts file (UNIX) or SQLHOSTS registry key (Windows) to make sure that you are using the value of the DBSERVERNAME configuration for the primary server correctly. Check if the value for the SDS_PAGING configuration parameter in the server configuration file (onconfig) is correct. Free some system resources and try to start the database server again.
177	The database server failed to start the main_loop thread.	Free some physical memory on the system and try to start the database server again.
178	The database server failed to initialize the memory required for page conversion.	Free some physical memory on the system and try to start the database server again.
179	The database server was unable to start CPU VPs.	Free some physical memory on the system and try to start the database server again.
180	The database server was unable to start the ADM VP.	Free some physical memory on the system and try to start the database server again.
181	The database server failed to initialize kernel AIO.	Free some physical memory on the system and try to start the database server again.
182	The database server was unable to start IO VPs.	Free some physical memory on the system and try to start the database server again.
183	The database server failed to initialize the memory required for asynchronous I/O operations.	Free some physical memory on the system and try to start the database server again.
184	The database server failed to initialize memory required for parallel database queries. (PDQ)	Free some physical memory on the system and try to start the database server again.

Table 12-1. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
185	The database server failed to initialize various SQL caches.	Free some physical memory on the system and try to start the database server again.
186	The database server failed to initialize the Global Language Support (GLS) component.	Free some physical memory on the system and try to start the database server again.
187	The database server failed to initialize the Associated Service Facility (ASF) components.	Check the entries in sqlhosts file.
188	The database server was unable to start the CRYPTO VP.	Free some physical memory on the system and try to start the database server again.
189	The database server was unable to initialize the alarm program.	Free some physical memory on the system and try to start the database server again.
190	The database server failed to initialize the auditing component.	Free some physical memory on the system and try to start the database server again.
192	The database server failed to restore the Window station and desktop.	(Windows only) Try to shut down the database server after freeing some system resources.
193	The database server failed to create daemon processes.	(UNIX and Mac OS only) Free some system resources and try to startup the database server once again.
194	The database server failed to redirect the file descriptors properly.	(UNIX and Mac OS only) Check the availability of the /dev/null device and try to start the database server again.
195	The database server failed to initialize the current directory for use.	Check the validity of the current working directory from where the database server is being initialized.
196	The database server failed to initialize the /dev/null device.	(AIX only) Check the validity of the /dev/null device.
197	The database server failed to find the password information for the user trying to initialize the database server.	Verify that the user password is valid.
198	The database server failed to set the resource limits.	(UNIX and Mac OS only) Verify, and if required, increase the resource limits for processes on the host computer.
200	The database server did not have enough memory to allocate structures during initialization.	Free some physical memory on the system and try to start the database server again.
206	The database server could not allocate the first resident segment.	Check the values of the BUFFERPOOL and LOCKS configuration parameters in the server configuration file (onconfig) to make sure that they can be accommodated with the available memory on the host computer.
207	The database server failed to initialize shared memory and disk space.	Free some physical memory in the system, check the validity of all the chunks in the database server, and try to start the database server again.
208	The database server failed to allocate structures from shared memory.	Free some system resources and try to start the database server again.
209	The database server encountered a fatal error during the creation of shared memory.	Free some physical memory in the system and try to start the database server again.
210	The database server requested memory for the resident segment that exceeded the maximum allowed.	Reduce the size of the resident segment by lowering the values of the BUFFERPOOL and LOCKS configuration parameters.
220	The database server failed to read the audit configuration file.	Check that the audit configuration file (adtcfg) exists and is valid.

Table 12-1. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
221	The database server could not detect the default directory for DUMPDIR. Usually it is the \$INFORMIXDIR/tmp directory.	Create the \$INFORMIXDIR/tmp directory if it is not present.
222	The database server detected an error in the value of the DBSERVERALIASES configuration parameter in the server's configuration file.	Verify that the values for the DBSERVERALIASES configuration parameter are valid and they have corresponding entries in the sqlhosts file (UNIX) or SQLHOSTS registry key (Windows).
223	The database server detected an error with the value of the DBSERVERNAME configuration parameter in the server's configuration file.	Verify that the value of the DBSERVERNAME configuration parameter is valid and it has a corresponding entry in the sqlhosts file (UNIX) or SQLHOSTS registry key (Windows).
224	The database server detected an error with the value of the HA_ALIAS configuration parameter in the server's configuration file.	Correct the value of the HA_ALIAS configuration parameter in the server configuration file (onconfig).
225	The database server detected too many entries for the NETTYPE configuration parameter or the DBSERVERALIASES configuration parameter in the server's configuration file.	Reduce the number of instances of the NETTYPE or DBSERVERALIASES configuration parameters in server configuration file (onconfig) and try to start the database server again.
226	The database server could not find an entry for the DBSERVERNAME configuration parameter in the sqlhosts file or the contents of the sqlhosts file are not valid.	Check the entries in the sqlhosts file.
227	Incorrect serial number.	Reinstall the database server.
228	The user does not have the necessary DBSA privileges to invoke the executable.	The user must have DBSA privileges or be a part of the Informix-Admin group (Windows).
229	The database server could not initialize the security sub-system.	(Windows only) The user does not the necessary user rights on the host or is not part of the Informix-Admin group.
230	The database server, if started as a process on Windows platform , timed out while trying to build the required system databases during initialization. (Windows only)	Check the event log on the host to determine why the service could not be opened or could not be started. The database server might have timed out while trying to build the system databases. Free some system resources and try to start the database server again.
231	Informix service startup failed when the "oninit -w" command was run as a process on the command line.	(Windows only) Check the event log on the host to determine why the service start has failed.
233	The database server failed to initialize the Pluggable Authentication Module (PAM).	Check the configuration for the PAM library on the system.
235	The database server detected errors for certain configuration parameter values in the server's configuration file.	Inspect the server configuration file (onconfig) for any errors.
236	The database server detected an error while trying to restrict the allowable values for the Informix edition in use.	Check if the SDS_ENABLE configuration parameter is set to 1 in the server configuration file (onconfig). Check if the server name specified with the oninit -SDS command matches the value of the HA_ALIAS or DBSERVERNAME configuration parameter. Check if the shared disk used is part of an existing shared disk cluster.
237	The database server could not find the server configuration file.	Ensure that the server configuration file exists and is valid.

Table 12-1. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
238	The database server detected an incorrect value for the INFORMIXSERVER environment variable or the value did not match the value of the DBSERVERNAME configuration parameter in the server's configuration file.	(Windows only) Check the value of the INFORMIXSERVER environment variable and the corresponding entry in the registry.
239	The database server detected an incorrect or non-existent value for the INFORMIXDIR environment variable.	(Windows only) Check the value of the INFORMIXDIR environment variable.
240	Incorrect command-line options were issued to the database server.	Correct the command-line options issued to the database server at startup.
248	The database server failed to create the Informix loader domain file.	(AIX only) Check if the /var/adm/ifx_loader_domain file is present.
249	The database server failed to dynamically load the PAM library.	The PAM library is not available for the database server. Install the PAM libraries.
250	The database server failed to dynamically load the ELF library.	The ELF library is not available to the database server. Install the libelf packages.
255	There was an internal error during server initialization. Look at any error messages written to stderr or to the online message log.	Take the appropriate action based on the error messages written to stderr or the online message log.

Chapter 13. The onlog Utility

The **onlog** utility displays the contents of a logical-log file, either on disk or on backup.

onlog: Display Logical-Log Contents

The **onlog** output is useful in debugging situations when you want to track a specific transaction or see what changes have been made to a specific tblspace. (For information about interpreting the logical-log file contents, see Chapter 5, “Interpreting Logical-Log Records,” on page 5-1.)

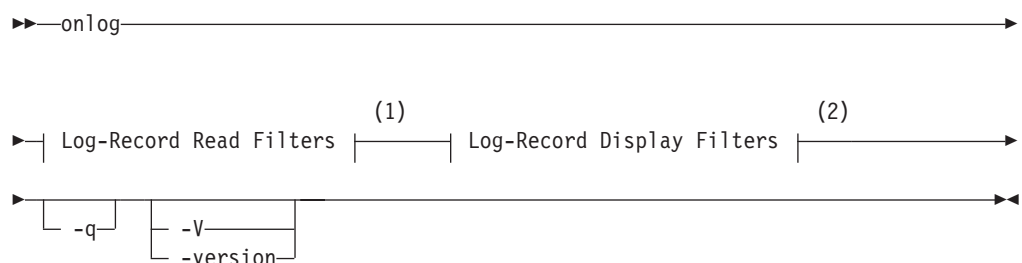
Any user can run all of the **onlog** options except the **-l** option. Only user **informix** on UNIX or a member of the **Informix-Admin** group on Windows can run the **-l** option.

If the database server is in offline mode when you execute **onlog**, only the files on disk are read. If the database server is in quiescent or online mode, **onlog** also reads the logical-log records stored in the logical-log buffers in shared memory (after all records on disk have been read).

When the database server reads a logical-log file with status U from disk while in online mode, the database server denies all access to the logical-log files, effectively stopping database activity for all sessions. (For more information, see “**onstat -l** command: Print physical and logical log information” on page 20-193.) For this reason, it is recommended that you wait until the files have been backed up and then read the contents of the logical-log files from backup.

The **onlog** utility does not have a functionally equivalent SQL administration API *command* string.

onlog Syntax



Notes:

- 1 see “Log-Record Read Filters” on page 13-2
- 2 see “Log-Record Display Filters” on page 13-3

Element	Purpose	Key Considerations
-q	Suppresses the initial header and the one-line header that appears every 18 records by default	None.

Element	Purpose	Key Considerations
-V	Displays the software version number and the serial number	See "Obtaining Utility Version Information" on page 6-1.
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See "Obtaining Utility Version Information" on page 6-1.

You direct **onlog** to read the following portions of the logical log as it searches for records to display:

- Records stored on disk
- Records stored on backup media
- Records from the specified logical-log file

By default, **onlog** displays the logical-log record header, which describes the transaction number and the record type. The record type identifies the type of operation performed.

In addition to the header, you can use the read filters to direct **onlog** to display the following information:

- Logical-log record header and data (including copies of simple large objects stored in a dbspace or tblspace)
- Copies of blobpages from blobspaces
They are copied from the logical-log backup only. They are not available from disk.

You can display every logical-log record header, or you can specify output based on the following criteria:

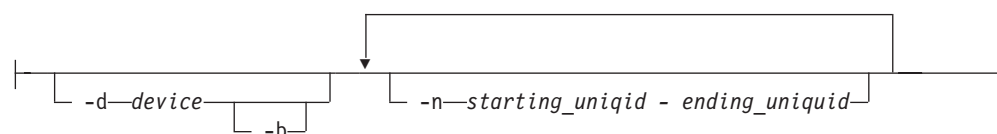
- Records associated with a specific table
- Records initiated by a specific user
- Records associated with a specific transaction

If **onlog** detects an error in the log file, such as an unrecognizable log type, it displays the entire log page in hexadecimal format and terminates.

Log-Record Read Filters

The **onlog** utility uses the pathnames that are stored in the root dbspace reserved pages to locate the logical-log files. If you use ON-Bar to back up the logical logs, **onlog** asks the storage manager to retrieve the appropriate logical-log records from the backup media.

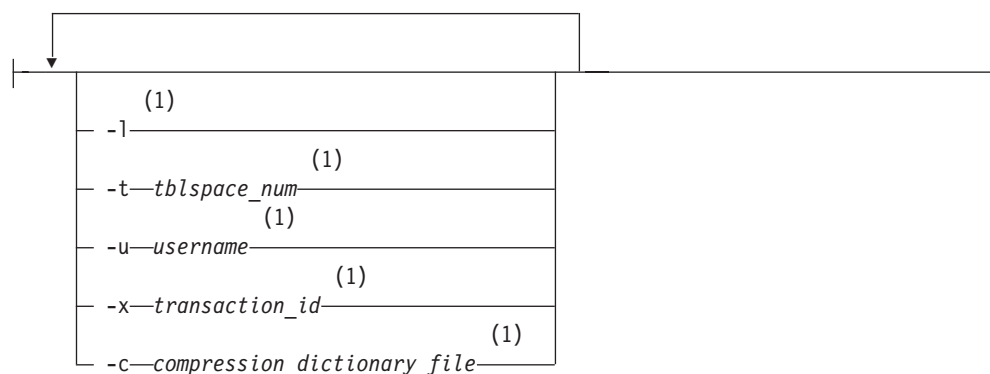
Syntax:



Element	Purpose	Key Considerations
-b	Displays logical-log records associated with blobSPACE blobpages	The database server stores these records on the logical-log backup media as part of blobSPACE logging.
-d device	Names the pathname of the storage device where the desired logical-log backup is mounted	<p>If you use ontape, the device that you name must be the same as the pathname of the device assigned to the configuration parameter LTAPEDEV. If the -d option is not used, onlog reads the logical-log files stored on disk, starting with the logical-log file with the lowest <i>logid</i>.</p> <p>If you use ON-Bar to back up logical logs, use the onbar -P command to view the contents of a logical-log file. See the <i>IBM Informix Backup and Restore Guide</i>.</p> <p>For pathname syntax, see your operating-system documentation.</p>
-n starting_uniqid- ending_uniqid	Directs onlog to read all the logical-log records contained in the log file that you specified from <i>starting uniqid</i> to the <i>ending uniqid</i> .	<p>The <i>starting_uniqid</i> and the <i>ending_uniqid</i> are the unique ID numbers of the logical log. To determine the <i>uniqid</i> of a particular logical-log file, use the onstat -l command.</p> <p>If you do not use the -n option, onlog reads all the logical-log files that are available (either on disk or on tape).</p> <p>For information about the onstat utility, see “Monitor the database server status” on page 20-18.</p>

Log-Record Display Filters

Syntax:



Notes:

- 1 Only one occurrence of this item allowed

Element	Purpose	Key Considerations
-l	Displays the long listing of the logical-log record.	The long listing of a log record includes a complex hexadecimal and ASCII dump of the entire log record. The listing is not intended for casual use.

Element	Purpose	Key Considerations
-t <i>tblspace_num</i>	Displays records associated with the tblspace that you specify.	Unsigned integer. Number, greater than 0, must be in the partnum column of the systables system catalog table. Specify this value as either an integer or hexadecimal value. (If you do not use a 0x prefix, the value is interpreted as an integer.) To determine the tblspace number of a particular tblspace, query the systables system catalog table as described in “Tblspace Numbers” on page 4-5.
-u <i>username</i>	Displays records for a specific user.	User name must be an existing login name. User name must conform to operating-system-specific rules for login name.
-x <i>transaction_id</i>	Displays only records associated with the transaction that you specify.	Value must be an unsigned integer between 0 and TRANSACTIONS - 1, inclusive. You should need to use the -x option only in the unlikely case that an error is generated during a rollforward. When this situation occurs, the database server sends a message to the message log that includes the transaction ID of the offending transaction. You can use this transaction ID with the -x option of onlog to investigate the cause of the error.
-c <i>compression_dictionary_file</i>	Uses the compression dictionary to expand compressed data and display uncompressed data.	If the onlog command contains the -l option and the -c option and there are compressed images in the log records, the onlog utility uses the compression dictionary to expand all expandable images in the log records. A compressed image is expandable only if there is a valid compression dictionary for that log record in the compression dictionary file. If -c is not specified or the compression dictionary file does not contain a valid compression dictionary for the compressed image, the onlog utility will display the row image in its compressed format.

If you do not have a compression dictionary file, you can use an UNLOAD statement to unload the compression dictionary, which is contained in the **syscompdicts_full** table in the **sysmaster** database, to a compression dictionary file, as follows:

```
UNLOAD TO 'compression_dictionary_file'
SELECT * FROM sysmaster:syscompdicts_full;
```

If you do not specify any options, **onlog** displays a short listing of all the records in the log. You can combine options with any other options to produce more selective filters. For example, if you use both the **-u** and **-x** options, **onlog** displays only the activities that the specified user initiated during the specified transaction. If you use both the **-u** and **-t** options, **onlog** displays only the activities initiated by the specified user and associated with the specified tblspace.

Related reference:

“alter logmode argument: Change the database logging mode (SQL administration API)” on page 21-20

Chapter 14. The **onmode** utility

Use the **onmode** utility to change the database server operating mode and perform various other operations on shared memory, sessions, transactions, parameters, and segments.

These topics show how to use the **onmode** options. If you do not use any options, the database server returns a usage statement.

On UNIX, you must be user **root** or user **informix** to run the **onmode** utility.

On Windows, you must be a member of the **Informix-Admin** group or the Administrators group to run the **onmode** utility.

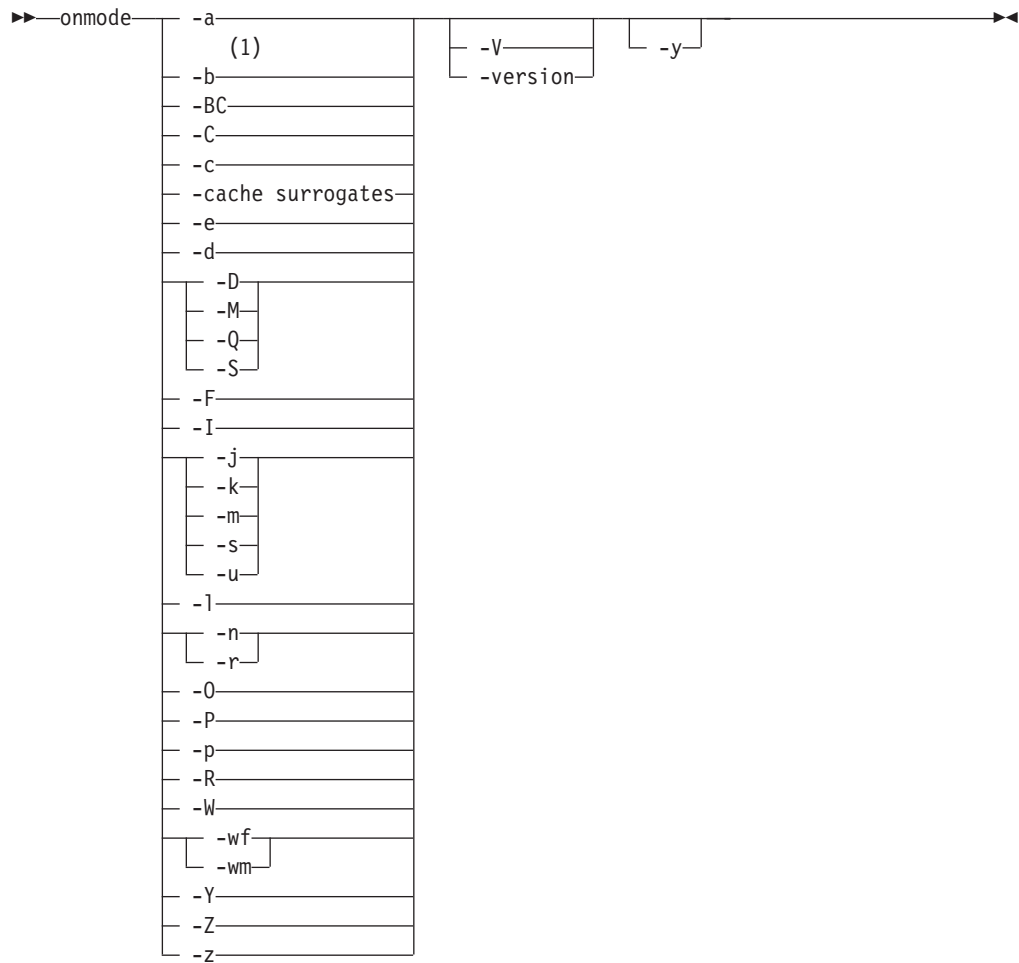
For information on the **onmode -b** command, which is only used if you upgraded to a new version of Informix and need to revert your databases to the previous version of the server, see Syntax of the **onmode -b** command in the *IBM Informix Migration Guide*.

All **onmode** command options have equivalent SQL administration API *command* strings, except **onmode -b**, **onmode -BC**, and **onmode -R**.

onmode Syntax

Use **onmode** utility commands to perform various database server operations.

The following syntax diagram shows all of the options that you can use with the **onmode** command. The syntax diagram does not show all of the elements that you use with each command option. For the complete syntax of each command, see the topic on that command.



Notes:

- 1 For a description of the **onstat -b** command, see the *IBM Informix Migration Guide*.

Element	Purpose	Key Considerations
-y	Causes the database server to automatically respond yes to all prompts	None.
-V	Displays the software version number and the serial number	See "Obtaining Utility Version Information" on page 6-1.
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See "Obtaining Utility Version Information" on page 6-1.

onmode -a: Add a shared-memory segment

Syntax

```
onmode -a seg_size
```

Element	Purpose	Key considerations
-a <i>seg_size</i>	Allows you to add a new virtual shared-memory segment. Size is specified in kilobytes	Restriction: The value of <i>seg_size</i> must be a positive integer. It must not exceed the operating system limit on the size of shared-memory segments.

Ordinarily, you do not need to add segments to the virtual portion of shared memory because the database server automatically adds segments as they are needed. However, as segments are added, the database server might reach the operating-system limit for the maximum number of segments before it acquires the memory that it needs. This situation typically occurs when the SHMADD configuration parameter is set so small that the database server exhausts the number of available segments before it acquires the memory that it needs for some operation.

You can use this command to add a segment that is larger than the size specified by the SHMADD configuration parameter. By using this command to add a segment, you can adhere to the operating system limit for segments while meeting the need that the database server has for additional memory.

This command has an equivalent SQL administration API function.

Related reference:

“add memory argument: Increase shared memory (SQL administration API)” on page 21-17

“onmode and a arguments: Add a shared-memory segment (SQL administration API)” on page 21-71

onmode -BC: Allow large chunk mode

Syntax:

```

>> onmode -BC 1
          -BC 2
  
```

Element	Purpose	Key Considerations
-BC 1	Enables support of large chunks, large offsets that are greater than 2 GB, and allows up to 32,768 chunks per instance.	This option allows large chunks to be created. Reversion without dropping the dbspace is possible if no chunks are larger than 2 GB. Dbspaces and blobspaces without chunks greater than 2 GB remain in the old format. After a chunk larger than 2 GB is added to a dbspace or blobspace then all chunks added or altered in that dbspace or blobspace are in the new format. See your <i>IBM Informix Administrator's Guide</i> .
-BC 2	Allows large-chunk-only mode for all dbspaces.	Reversion is not possible. Enables the 9.4 large chunk feature for all dbspaces and blobspaces, Any chunk or offset added or modified has the new format. Existing chunks that you do not alter remain in the old format. See your <i>IBM Informix Administrator's Guide</i>

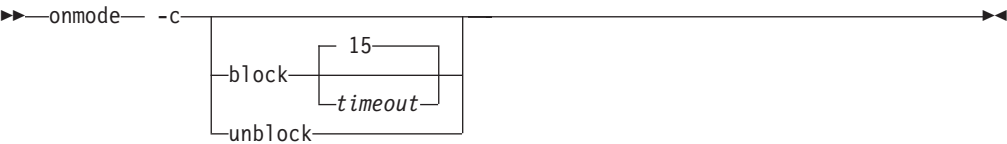
The **onmode -BC** (backward-compatible) commands are useful if you have converted from Informix 9.40 (small chunk mode) to Informix 10.0 or later. When Informix 10.0 or later is first initialized (with the **oninit -iyv** command), by default

it comes online with large chunk mode already fully enabled. Reversion is not possible. In the case of a newly initialized instance of Informix 10.0 or later, the **onmode -BC** commands will return an error.

Note: After executing the **onmode -BC** command, perform a complete system level-0 backup.

onmode -c: Force a checkpoint

Syntax:



Element	Purpose	Key considerations
-c	Forces a checkpoint that flushes the buffers to disk.	You can use the -c option to force a sync checkpoint if the most recent checkpoint record in the logical log was preventing the logical-log file from being freed (status U-B-L).
block	Blocks the database server from any transactions.	While the database server is blocked, users can access it in read-only mode. Use this option to perform an external backup on IBM Informix. For more information, see the <i>IBM Informix Backup and Restore Guide</i> .
timeout	Specifies the number of seconds to wait for checkpoints to clear before returning to the command prompt.	The <i>timeout</i> option applies only if the DELAY_APPLY configuration parameter is configured (see “DELAY_APPLY Configuration Parameter” on page 1-53. If the DELAY_APPLY configuration parameter is enabled, the checkpoint requested by the primary server might not arrive at the secondary server for an extended period of time. It is also possible that no other checkpoints are staged in the staging directory. The default timeout value is 15 seconds and the maximum timeout allowed is 10 minutes (600 seconds). See <i>IBM Informix Backup and Restore Guide</i> .
unblock	Unblocks the database server.	When the database server is unblocked, data transactions and normal database server operations can resume. Use this option after you complete an external backup on IBM Informix. For more information, see the <i>IBM Informix Backup and Restore Guide</i> .

This command has an equivalent SQL administration API function.

Related reference:

“checkpoint argument: Force a checkpoint (SQL administration API)” on page 21-25

“onmode and c arguments: Force a checkpoint (SQL administration API)” on page 21-72

onmode -C: Control the B-tree scanner

Syntax:

```

▶▶ onmode -C (yielding syntax)
    start
    count
    stop count
    kill count
    threshold size
    duration num
    rangesize size
    alice mode
    compression value
  
```

Element	Purpose	Key considerations
-C	Controls the B-tree scanner for cleaning indexes of deleted items	There is no limit to the number of threads that can run at one time. However, there is a limit of 128 threads that can be started at one time. If, for example, you wanted 150 threads to run, you could execute two commands: onmode -C 100 and onmode -C 50 .
start count	Starts additional B-tree scanner threads.	If <i>count</i> is not specified, a <i>count</i> of 1 is assumed. There is no limit on the number of scanner threads that can be specified.
stop count kill count	Stops B-tree scanner threads.	If <i>count</i> is not specified, a <i>count</i> of 1 is assumed. Stopping all index scanners prevents all index cleaning. Either of these commands stop the B-tree scanner.
threshold size	Sets the minimum number of deleted items an index must encounter before an index is placed on the hot list.	Once all indexes above the threshold have been cleaned and there is no other work for the B-tree scanner to do, the indexes below the threshold are added to the hot list.
duration num	The number of seconds that the hot list is valid.	After this number of seconds expires, the hot list will be rebuilt by the next available B-tree scanner thread, even if unprocessed items are on the list. Scanners currently processing requests are not interrupted.
rangesize size	Determines the size of an index before index range cleaning is enabled.	A size of -1 can be used to disable range scanning.
alice num	Sets the system's alice mode.	Valid <i>num</i> values range from 0 (OFF) to 12.
compression value	For a database server instance, modifies the level at which two partially used index pages are merged. The pages are merged if the data on those pages totals a set level.	Valid values for the level are low, med (medium), high, and default. The system default value is med.

The B-tree scanner has statistical information which tracks index efficiency and how much extra work the index currently places on the server. Based on the

amount of extra work the index has accomplished because of committed deleted index items, the B-tree scanner develops an ordered list of indexes that have caused the server to do extra work. This list is called the hot list. The index causing the highest amount of extra work is cleaned first and the rest of the indexes are cleaned in descending order. The DBA can allocate cleaning threads dynamically, thus allowing for configurable workloads.

This command has an equivalent SQL administration API function.

Related reference:

“onmode and C arguments: Control the B-tree scanner (SQL administration API)” on page 21-73

“set index compression argument: Change index page compression (SQL administration API)” on page 21-106

onmode -cache surrogates: Cache the allowed.surrogates file

Syntax:

►► onmode — -cache surrogates —————►►

Element	Purpose	Key considerations
-cache surrogates	Reads the /etc/informix/ allowed.surrogates file and stores the user IDs and group IDs values in shared memory cache. The user names and group names specified in allowed.surrogates file have to be valid operating system users and groups. The names are converted to corresponding UIDs and GIDs.	<p>You can use onmode -cache surrogates during a session to load the allowed.surrogates file. The allowed.surrogates file is used specify users and groups who can act as surrogates for mapped users. The allowed.surrogates file will be automatically checked before a new connection is made to the database server or when users are created or altered.</p> <p>If the cache-refresh fails, the existing surrogate cache is cleared, effectively disabling mapped users. Existing connections on the server will be unaffected by changes in shared-memory cache. Changes in shared memory cache affect new sessions.</p>

Related concepts:

➡ Mapped user surrogates in the allowed.surrogates file (UNIX, Linux) (Security Guide)

Related tasks:

➡ Specifying surrogates for mapped users (UNIX, Linux) (Security Guide)

Related reference:

➡ Surrogate user properties (SQL Syntax)

onmode -d: Set data-replication types

Syntax:

►► onmode — -d ————►►
 └── standard —————►►
 └── primary — dbservername —►►
 └── secondary —►►

Element	Purpose	Key Considerations
-d	Used to set the High-Availability Data-Replication type, either standard, primary, or secondary, as the following sections describe	You can use the -d primary and -d secondary options only when the database server is in quiescent mode. You can use the -d standard option when the database server is in quiescent, online, or read-only mode.
<i>dbservername</i>	Identifies the database server name of the primary or secondary database server	<p>The <i>dbservername</i> argument must correspond to the DBSERVERNAME parameter in the ONCONFIG file of the intended secondary database server. It should <i>not</i> correspond to one of the database servers that the DBSERVERALIASES parameter specifies.</p> <p>The <i>dbservername</i> argument of the other database server in the data-replication pair and the type of a database server (standard, primary, or secondary) is preserved after reinitialization of shared memory.</p> <p>For more information, see <i>range of values</i> for the DBSERVERNAME configuration parameter in “DBSERVERNAME configuration parameter” on page 1-47.</p>

Related reference:

“ha set primary argument: Define an HDR primary server (SQL administration API)” on page 21-60

“ha set secondary argument: Define an HDR secondary server (SQL administration API)” on page 21-61

“ha set standard argument: Convert an HDR server into a standard server (SQL administration API)” on page 21-61

“onmode and d arguments: Set data-replication types (SQL administration API)” on page 21-74

Using the -d standard Option

The **-d standard** option drops the connection between database servers in a data replication pair (if one exists) and sets the database server type of the current database server to standard. This option does not change the mode or type of the other database server in the pair.

Use the **onmode -d standard** command only to disconnect a primary server from an HDR secondary server. Running the command converts the HDR secondary server to a standalone server. You should not run the **onmode -d standard** command to disconnect a primary server from an RS secondary server. To disconnect a primary server from an RS secondary server run the following commands:

On the primary server:

```
onmode -d delete RSS dbservername
```

On the RS secondary server:

```
onmode -d standard
```

Using the -d primary dbservername Option

The **-d primary *dbservername*** option sets the database server type to primary and attempts to connect with the database server that *dbservername* specifies. If the

connection is successful, data replication is turned on. The primary database server goes into online mode, and the secondary database server goes into read-only mode. If the connection is not successful, the database server comes to on-line mode, but data replication is not turned on.

Using the -d secondary dbservername option

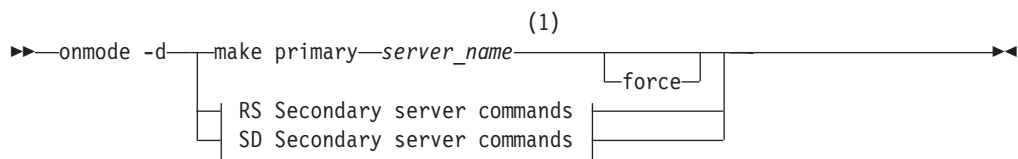
The **-d secondary dbservername** option sets the database server type to secondary and attempts to connect with the database server that **dbservername** specifies. If the connection is successful, data replication is turned on. The primary database server goes online, and the secondary database server goes into read-only mode. If the connection is not successful, the database server comes to read-only mode, but data replication is not turned on.

This command has an equivalent SQL administration API function. For more information see “SQL Administration API Overview” on page 21-1 and the *IBM Informix Administrator’s Guide*.

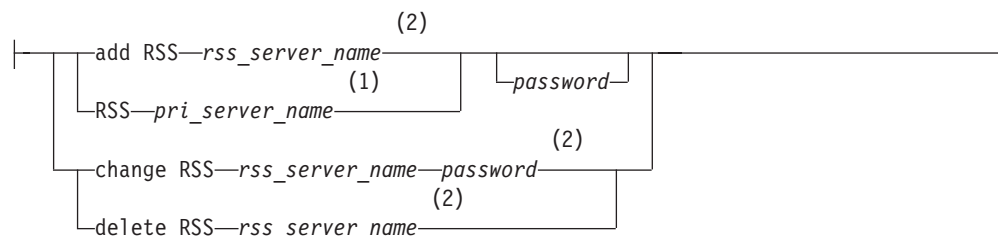
For other **onstat -d** information, see “onmode -d: Set High Availability server characteristics” and “**onmode -d** command: Replicate an index with data-replication” on page 14-10.

onmode -d: Set High Availability server characteristics

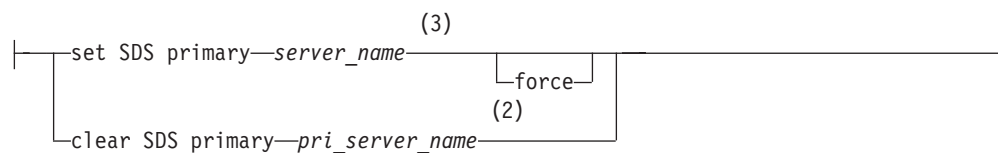
Syntax:



RS Secondary server commands:



SD Secondary server commands:



Notes:

- 1 Run on secondary server only.

- 2 Run on primary server only.
- 3 Run on primary server or secondary server.

Element	Purpose	Key Considerations
-d	Used to create, modify, or delete secondary servers in high-availability configurations	
add RSS	Adds an RS secondary server	This command should be run on the primary database server.
<i>rss_server_name</i>	Identifies the RS secondary database server name	The <i>servername</i> argument can be DBSERVERNAME, DBSERVERALIASES and ER group name. For more information, see <i>range of values</i> for DBSERVERNAME configuration parameter in “DBSERVERNAME configuration parameter” on page 1-47 and DBSERVERALIASES in “DBSERVERALIASES configuration parameter” on page 1-45.
<i>password</i>	Specifies the secondary server password	The password is used only during the first connection attempt. After the primary and secondary server have connected, the password cannot be changed.
RSS	Sets an RS secondary server type	This command should be run on the secondary database server.
<i>pri_server_name</i>	Identifies the name of the primary server	
change RSS	Change an RS secondary server	This command should be run on the primary database server.
delete RSS	Removes an RS secondary server definition	This command should be run on the primary database server.
set SDS primary	Defines the server as a shared disk primary server	
<i>server_name</i>	The name of the database server	When used with set SDS or make primary , this is the name of the server whose role is changing.
force	Used to force a change	If the force option is specified, the operation is performed without requiring that the secondary server be connected to the current primary server. If the force option is not specified, the operation must be coordinated with the current primary server. The force option should be used only when the DBA is certain that the current primary server is not active; otherwise, the shared disk subsystem can become corrupted.
clear SDS primary	Disables the shared disk environment. The server name specified no longer acts as an SD primary server	
make primary	Creates a primary server	The make primary command can be issued on any type of secondary server, including HDR secondary, RS secondary, and SD secondary servers. If make primary is run on: <ul style="list-style-type: none"> • HDR Secondary: The current primary server is shut down and the secondary is made the primary. • RS secondary: The server is changed to a standard server. • SD secondary: The server is made the new primary server.

You can also set data replication characteristics can with SQL administration API *command* equivalents. For more information see “SQL Administration API Overview” on page 21-1 and the *IBM Informix Administrator’s Guide*.

For other **onmode -d** information, see “onmode -d: Set data-replication types” on page 14-6 and “**onmode -d** command: Replicate an index with data-replication.”

Related reference:

“ha make primary argument: Change the mode of a secondary server (SQL administration API)” on page 21-52

“ha rss argument: Create an RS secondary server (SQL administration API)” on page 21-53

“ha rss add argument: Add an RS secondary server to a primary server (SQL administration API)” on page 21-54

“ha rss change argument: Change the password of an RS secondary server (SQL administration API)” on page 21-55

“ha rss delete argument: Delete an RS secondary server (SQL administration API)” on page 21-56

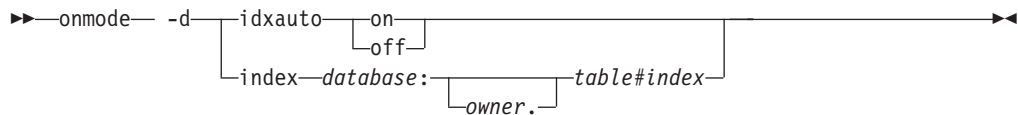
“ha sds clear argument: Stop shared-disk replication (SQL administration API)” on page 21-56

“ha sds set argument: Create a shared-disk primary server (SQL administration API)” on page 21-58

“ha sds primary argument: Convert an SD secondary server to a primary server (SQL administration API)” on page 21-57

onmode -d command: Replicate an index with data-replication

Syntax:



Element	Purpose	Key considerations
-d	Specifies how indexes are replicated to a High-Availability Data-Replication (HDR) secondary server when an index on the secondary server becomes corrupt	You can use the onmode -d idxauto and onmode -d index commands while the server is in online mode.
idxauto	Enables automatic index replication when an index on a secondary server becomes corrupt	Use the onmode -d idxauto command to overwrite the value of the DRIDXAUTO configuration parameter within a session. For more information on DRIDXAUTO, see “DRIDXAUTO Configuration Parameter” on page 1-59. For more information on replicating indexes, see the chapter on using High-Availability Data Replication in the <i>IBM Informix Administrator’s Guide</i> .
index	Replicates an index from a primary to a secondary server	If you detect a corrupt index on a secondary server, use the onmode -d index command to start replication of the index from the primary to the secondary server.

Element	Purpose	Key considerations
<i>database</i>	Specifies the database containing the index to replicate	Syntax must conform to the Identifier segment; see the <i>IBM Informix Administrator's Guide</i> .
<i>index</i>	Specifies the name of the index to replicate	Index must exist on table and in database specified. Syntax must conform to the Identifier segment; see the <i>IBM Informix Administrator's Guide</i> .
<i>owner</i>	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Table Name segment; see the <i>IBM Informix Administrator's Guide</i> .
<i>table</i>	Specifies the name of the table on which the index is based	Syntax must conform to the Table Name segment; see the <i>IBM Informix Administrator's Guide</i> .

The **onmode -d idxauto** and the **onmode -d index** commands provide methods to replicate an index to a secondary server containing a corrupted index. The base table will be locked during the transfer of an index. The alternative to using these options is to drop and rebuild the corrupt index on the primary server.

In the case of a fragmented index with one corrupt fragment, the **onmode -d idxauto** command only transfers the single affected fragment, whereas the **onmode -d index** command transfers the whole index.

This command has an equivalent SQL administration API function. For more information see “SQL Administration API Overview” on page 21-1 and the *IBM Informix Administrator's Guide*.

For additional **onmode -d** command information, see “onmode -d: Set High Availability server characteristics” on page 14-8 and “onmode -d: Set data-replication types” on page 14-6.

Related reference:

“ha set idxauto argument: Replicate indexes to secondary servers (SQL administration API)” on page 21-59

onmode -D, -M, -Q, -S: Change decision-support parameters

Syntax:

```

>> onmode -D max_priority -M kilobytes -Q queries -S scans

```

Element	Purpose	Key considerations
-D max_priority	Changes the value of MAX_PDQPRIORITY	This value must be an unsigned integer between 0 and 100. Specify <i>max_priority</i> as a factor to temper user requests for PDQ resources. For information on parameters used for controlling PDQ, see “MAX_PDQPRIORITY Configuration Parameter” on page 1-94 and the <i>IBM Informix Performance Guide</i> .

Element	Purpose	Key considerations
-M kilobytes	Changes the value of DS_TOTAL_MEMORY	<p>This value has a platform-dependent upper limit. If you enter a very large value and that value is too large for your platform, you will receive a message that gives you the range of values for your platform.</p> <p>Specify <i>kilobytes</i> for the maximum amount of memory available for parallel queries.</p> <p>For more information, see “DS_TOTAL_MEMORY configuration parameter” on page 1-64 and the <i>IBM Informix Performance Guide</i>.</p>
-Q queries	Changes the value of DS_MAX_QUERIES	<p>This value must be an unsigned integer between 1 and 8,388,608.</p> <p>Specify <i>queries</i> for the maximum number of concurrently executing parallel queries.</p> <p>For information on parameters used for controlling PDQ, see “DS_MAX_QUERIES Configuration Parameter” on page 1-61 and the <i>IBM Informix Performance Guide</i>.</p>
-S scans	Changes the value of DS_MAX_SCANS	<p>This value must be an unsigned integer between 10 and 1,048,576.</p> <p>Specify <i>scans</i> for the maximum number of concurrently executing parallel scans.</p> <p>For information on parameters used for controlling PDQ, see “DS_MAX_SCANS Configuration Parameter” on page 1-62 and the <i>IBM Informix Performance Guide</i>.</p>

These options allow you to change configuration parameters while the database server is online. The new values affect only the current instance of the database server; the values are not recorded in the ONCONFIG file. If you shut down and restart the database server, the values of the parameters revert to the values in the ONCONFIG file. For more information about these configuration parameters, see Chapter 1, “Database configuration parameters,” on page 1-1.

To check the current values for the MAX_PDQPRIORITY, DS_TOTAL_MEMORY, DS_MAX_SCANS, DS_MAX_QUERIES, and the DS_NONPDQ_QUERY_MEM configuration parameters, use **onstat -g mgm**. See “**onstat -g mgm** command: Print MGM resource information” on page 20-116.

This command has an equivalent SQL administration API function.

Related reference:

“onmode and D arguments: Set PDQ priority (SQL administration API)” on page 21-76

“onmode and M arguments: Temporarily change decision-support memory (SQL administration API)” on page 21-80

“onmode and Q arguments: Set maximum number for decision-support queries (SQL administration API)” on page 21-83

“onmode and S arguments: Set maximum number of decision-support scans (SQL administration API)” on page 21-85

onmode -e: Change usage of the SQL statement cache

Syntax:

►► onmode — -e mode —►►

Element	Purpose	Key considerations
onmode -e ENABLE	Enables the SQL statement cache. For more information, see the material on improving query performance in the <i>IBM Informix Performance Guide</i> .	User sessions use the cache only when they perform either of the following actions: <ul style="list-style-type: none"> Set the environment variable STMT_CACHE to 1 Execute the SQL statement SET STATEMENT CACHE ON
onmode -e FLUSH	Flushes the statements that are not in use from the SQL statement cache	The onstat -g ssc ref_cnt field shows 0.
onmode -e OFF	Turns off the SQL statement cache	No statements are cached.
onmode -e ON	Turns on the SQL statement cache	All statements are cached unless the user turns it off with one of the following actions: <ul style="list-style-type: none"> Set the environment variable STMT_CACHE to 0 Execute the SQL statement SET STATEMENT CACHE OFF

The **onmode -e** changes are in effect for the current database server session only. When you restart the database server, it uses the default STMT_CACHE parameter value in the ONCONFIG file.

This command has an equivalent SQL administration API function.

Related reference:

“onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)” on page 21-76

onmode -F: Free unused memory segments

Syntax:

►► onmode — -F —►►

Element	Purpose	Key considerations
-F	Frees unused memory segments	None.

When you execute **onmode -F**, the memory manager examines each memory pool for unused memory. When the memory manager locates blocks of unused memory, it immediately frees the memory. After the memory manager checks each memory pool, it begins checking memory segments and frees any that the database server no longer needs.

It is recommended that you run **onmode -F** from an operating-system scheduling facility regularly and after the database server performs any function that creates additional memory segments, including large index builds, sorts, or backups.

Running **onmode -F** causes a significant degradation of performance for any users that are active when you execute the utility. Although the execution time is brief (1 to 2 seconds), degradation for a single-user database server can reach 100 percent. Systems with multiple CPU virtual processors experience proportionately less degradation.

To confirm that **onmode** freed unused memory, check your message log. If the memory manager frees one or more segments, it displays a message that indicates how many segments and bytes of memory were freed.

This command has an equivalent SQL administration API function.

Related reference:

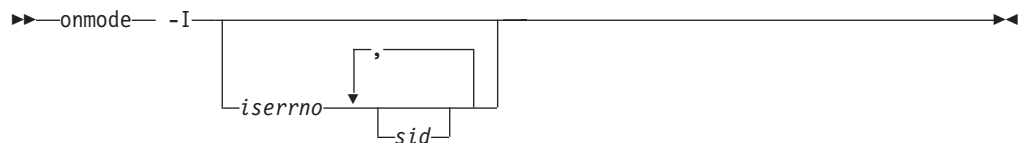
“onmode and F arguments: Free unused memory segments (SQL administration API)” on page 21-77

onmode -I: Control diagnostics collection

Use the **onmode -I** option to start and stop diagnostics collection.

When you encounter an error, you can specify the **onmode -I iserrno** option to start collecting diagnostics information. You can also specify the session ID to collect information for only specific session.

To stop the diagnostics collection, use the **onmode -I** option without any other parameters.



Element	Purpose	Key Considerations
<i>iserrno</i>	Message number of the error that you want to collect diagnostic information for.	None.
<i>sid</i>	Session ID of the session that you want to collect diagnostic information for.	None.

The diagnostics collection procedures are described in the *IBM Informix Administrator's Guide*.

onmode -k, -m, -s, -u, -j: Change database server mode

Syntax:



Element	Purpose	Key Considerations
-k	Takes the database server to offline mode and removes shared memory	To reinitialize shared memory, shut down and restart the database server. "Taking the Database Server to Offline Mode with the -k Option" on page 14-16.
-m	Takes the database server from quiescent or administration mode to online mode	See "Bringing the Database Server Online with the -m Option" on page 14-16.
-s	Shuts down the database server gracefully	Users who are using the database server are allowed to finish before the database server comes to quiescent mode, but no new connections are allowed. When all processing is finished, -s takes the database server to quiescent mode. The -s option leaves shared memory intact. See "Shutting Down the Database Server Gracefully with the -s Option" on page 14-16.
-u	Shuts down the database server immediately	This option brings the database server to quiescent mode without waiting for users to finish their sessions. Their current transactions are rolled back, and their sessions are terminated. See "Shutting Down the Database Server Immediately with the -u Option" on page 14-16.
-j	Puts the database server into administration mode	This option brings the database server to administration mode, allowing the informix user all functions including the issuance of SQL and DDL commands. The -j -U option enables the DBSA to designate specific users (in addition to the informix user) to access the database server. See your <i>IBM Informix Administrator's Guide</i> .

The following sections describe the options that take the database server from one mode to another.

Related reference:

“onmode and j arguments: Switch the database server to administration mode (SQL administration API)” on page 21-78

“onmode and m arguments: Switch to multi-user mode (SQL administration API)” on page 21-79

Taking the Database Server to Offline Mode with the -k Option

The **onmode -k** option takes the database server to offline mode and removes database server shared memory.

A prompt asks for confirmation. Another prompt asks for confirmation to kill user threads before the database server comes offline. If you want to eliminate these prompts, execute the **-y** option with the **-s** option.

This option does not kill all client sessions. Use the **-u** option to avoid hanging client sessions or virtual server processes.

Important: When you use the **onmode -k** command to shut down the database server, utilities that are waiting for a user response might not terminate. For example, **ontape** might be waiting for another tape, **onstat -i** might be waiting for a user response, or **onspaces** might be waiting for **y** or **n** to continue. If this problem occurs, use **onmode -uk** or **-uky** instead to roll back work before removing shared memory. For more information, see the descriptions of other options on this page.

Related reference:

Chapter 9, “The onclean Utility,” on page 9-1

“The onshutdown script” on page 9-2

Bringing the Database Server Online with the -m Option

The **-m** option brings the database server online from quiescent mode.

Shutting Down the Database Server Gracefully with the -s Option

The **-s** option causes a graceful shutdown. Users who are using the database server are allowed to finish before the database server comes to quiescent mode, but no new connections are allowed. When all processing is finished, **-s** takes the database server to quiescent mode. The **-s** option leaves shared memory intact.

A prompt asks for confirmation. If you want to eliminate this prompt, execute the **-y** option with the **-s** option.

Shutting Down the Database Server Immediately with the -u Option

The **-u** option causes immediate shutdown. This option brings the database server to quiescent mode without waiting for users to finish their sessions. Their current transactions are rolled back, and their sessions are terminated.

A prompt asks for confirmation. Another prompt asks for confirmation to kill user threads before the database server comes to quiescent mode. If you want to eliminate these prompts, execute the **-y** option with the **-s** option.

Changing the Database Server to Administration Mode with the -j Option

The **-j** option puts the database server into the administration mode and allows only the DBSA group and the user **informix** to connect to the server. The **-j** option allows a DBSA to have the server in a fully functional mode to perform maintenance.

The **-j -U** option enables the DBSA to grant individual users access to the database server in administration mode. Once connected, these individual users can execute any SQL or DDL command. When the server is changed to administration mode, all sessions for users other than user **informix**, the DBSA group users, and those identified in the **onmode -j -U** command lose their database server connection.

The following example enables three individual users to connect to the database server and have database server access until the database server mode changes to offline, quiescent or online mode:

```
onmode -j -U karin,sarah,andy
```

Access for individual users can also be removed by executing **onmode -j -U** and removing their name from the new list of names in the command. For example, in the following commands, the first command grants only Karin access, the second command grants Karin and Sarah access, and the third command grants only Sarah access (and removes access from Karin).

```
onmode -j -U karin
onmode -j -U karin,sarah
onmode -j -U sarah
```

To allow user **informix** and the DBSA group user to retain their database server access in administration mode and remove all single users from accessing the database server, use the following command:

```
onmode -j -U ' '
```

For information on designating single users in administration mode using a configuration parameter, see “ADMIN_MODE_USERS Configuration Parameter” on page 1-25

Related reference:

“ADMIN_MODE_USERS Configuration Parameter” on page 1-25

Changing Database Server Mode with ON-Monitor (UNIX)

You can also use ON-Monitor options to change the database server mode. The following table shows ON-Monitor options that are equivalent to the **onmode** options.

onmode Option	ON-Monitor Option
-k	Take-Offline
-m	On-Line
-s	Graceful-Shutdown
-u	Immediate-Shutdown
-j	Administration Mode

onmode -l: Switch the logical-log file

Syntax:

►► onmode — -l —►►

Element	Purpose	Key considerations
-l	Switches the current logical-log file to the next logical-log file	You must use onmode to switch to the next logical-log file. For information on switching to the next logical-log file, see the chapter on managing logical-log files in the <i>IBM Informix Administrator's Guide</i> .

This command has an equivalent SQL administration API function.

Related reference:

"onmode and l arguments: Switch to the next logical log (SQL administration API)" on page 21-79

onmode -n, -r: Change shared-memory residency

Syntax:

►► onmode — -n
-r —►►

Element	Purpose	Key considerations
-n	Ends forced residency of the resident portion of shared memory	This command does not affect the value of RESIDENT, the forced-residency parameter in the ONCONFIG file.
-r	Starts forced residency of the resident portion of shared memory	This command does not affect the value of RESIDENT, the forced-memory parameter in the ONCONFIG file.

Important: Set the RESIDENT parameter to 1 before you use the **onmode -r** or **-n** options.

For information on using the forced-residency parameter to turn residency on or off for the next time that you restart the database server, see the chapter on managing shared memory in the *IBM Informix Administrator's Guide*.

This command has an equivalent SQL administration API function.

Related reference:

“onmode and n arguments: Unlock resident memory (SQL administration API)” on page 21-81

“onmode and r arguments: Force residency of shared memory (SQL administration API)” on page 21-84

onmode -O: Override ONDBSPACEDOWN WAIT mode

Syntax:

►► onmode — -O ◀◀

Element	Purpose	Key considerations
-O	Overrides the WAIT mode of the ONDBSPACEDOWN configuration parameter	None.

Use the **onmode -O** option only in the following circumstances:

- ONDBSPACEDOWN is set to WAIT.
- A disabling I/O error occurs that causes the database server to block all updating threads.
- You cannot or do not want to correct the problem that caused the disabling I/O error.
- You want the database server to mark the disabled dbspace as down and continue processing.

When you execute this option, the database server marks the dbspace responsible for the disabling I/O error as down, completes a checkpoint, and releases blocked threads. Then **onmode** prompts you with the following message:

```
This will render any dbspaces which have incurred disabling I/O errors unusable
and require them to be restored from an archive.
Do you wish to continue?(y/n)
```

If **onmode** does not find any disabling I/O errors on noncritical dbspaces when you run the -O option, it notifies you with the following message:

```
There have been no disabling I/O errors on any noncritical dbspaces.
```

This command has an equivalent SQL administration API function.

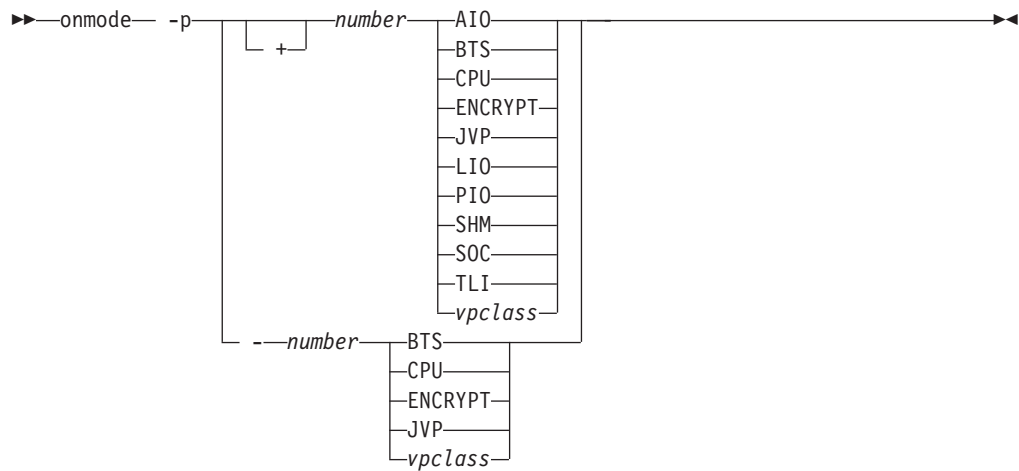
Related reference:

“onmode and O arguments: Mark a disabled dbspace as down (SQL administration API)” on page 21-81

onmode -p: Add or remove virtual processors

Use the **onmode -p** command to dynamically add or remove virtual processors for the current database session. The **onmode -p** command does not update the onconfig file.

Syntax:



Element	Purpose	Key Considerations
-p number	<p>Adds or removes virtual processors. The <i>number</i> argument indicates the number of virtual processors to add or remove</p> <p>If this value is a negative integer, processors are removed. If this value is a positive integer, processors are added.</p>	<p>You can use the -p option only when the database server is in online mode, and you can add to only one class of virtual processors at a time.</p> <p>For more details, see “Adding and Dropping Virtual Processors” on page 14-21.</p> <p>If you are removing virtual processors, the maximum cannot exceed the actual number of processors of the specified type. If you are adding virtual processors, the maximum number depends on the operating system.</p> <p>For more information, see the chapter on using virtual processors in the <i>IBM Informix Administrator's Guide</i>.</p>
AIO	Performs nonlogging disk I/O to cooked disk spaces	Also performs nonlogging I/O to raw disk spaces if kernel asynchronous I/O (KAIO) is not used.
BTS	Run basic text search index operations and queries.	<p>BTS virtual processors are non-yielding. Specify more BTS virtual processors if you want multiple basic text search queries to be run simultaneously. Use the VPCLASS parameter in the onconfig file to create at least one permanent BTS virtual processor.</p> <p>For more information on basic text search queries, see the <i>IBM Informix Database Extensions User's Guide</i>.</p>
CPU	Runs all session threads and some system threads	It is recommended that the number of CPU VPs not be greater than the number of physical processors. If KAIO is used, performs I/O to raw disk spaces, including I/O to physical and logical logs. Runs thread for KAIO where available or a single poll thread. The database server uses the number of CPU VPs to allocate resources for parallel database queries (PDQ). If you drop CPU VPs, your queries will run significantly slower. The Reinit field of the onstat -g mgm output displays information on the number of queries that are waiting for running queries to complete after an onmode -p command. Also see the <i>IBM Informix Performance Guide</i> .
ENCRYPT	Executes column-level encryption and decryption routines	Specify more ENCRYPT virtual processors if you have multiple encrypted columns.
JVP	Executes Java user-defined routines in the Java Virtual Machine (JVM)	Specify more JVPs if you are running many Java UDRs.

Element	Purpose	Key Considerations
LIO	Writes to the logical-log files if they are in cooked disk space	Use two LIO virtual processors only if the logical logs are in mirrored dbspaces. The database server allows a maximum of two LIO virtual processors.
PIO	Writes to the physical log if it is in cooked disk space	Use two PIO virtual processors only if the physical log is in a mirrored dbspace. The database server allows a maximum of two PIO virtual processors.
SHM	Performs shared-memory communication	You can use the SHM virtual processor even if the database server is not configured for shared-memory communication.
SOC	Uses sockets to perform network communications	You can use the SOC virtual processor only if the database server is configured for network connections through sockets.
STR	Performs stream pipe connections	
TLI	Uses the Transport Layer Interface (TLI) to perform network communication	You can use the TLI virtual processor only if the database server is configured for network connections through TLI.
<i>vpclass</i>	Names a user-defined virtual processor class	<p>Use the VPCLASS parameter in the <code>onconfig</code> to define the user-defined virtual-processor class. Specify more user-defined virtual processors if you are running many UDRs.</p> <p>On Windows, you can have only one user-defined virtual processor class at a time. Omit the <i>number</i> parameter in the onmode -p <i>vpclass</i> command.</p> <p>For more information on extension classes, see “VPCLASS configuration parameter” on page 1-156.</p>

Related reference:

“onmode and p arguments: Add or remove virtual processors (SQL administration API)” on page 21-82

Adding and Dropping Virtual Processors

The following rules about adding or dropping virtual processors apply:

- You can add but not drop virtual processors of the AIO, PIO, LIO, TLI, SHM, SOC, and STR classes.
- You cannot add or drop virtual processors of the OPT, ADM, ADT, and MSC classes. The database server adds them automatically.
- You can add or drop virtual processors of the CPU, ENCRYPT, JVP, and user-defined (*vpclass*) classes.
-

Windows Only: You can add a virtual processor of any class, but you cannot drop virtual processors.

Dropping Virtual Processors Automatically

Table 14-1 on page 14-22 shows the virtual processors that the database server starts automatically. You cannot add or drop these virtual processors with the **onmode -p** command. To drop these virtual processors, shut down and restart the database server.

Table 14-1. Virtual-Processor Classes That the Database Server Starts Automatically

Virtual-Processor Class	Description
ADM	Performs administrative functions
ADT	Runs auditing processes The database server starts one virtual processor in the audit class when you turn on audit mode by setting the ADTMODE parameter in the ONCONFIG file.
MSC	Services requests for system calls that require a large stack The database server starts this virtual processor automatically.
OPT	Performs I/O to the optical disk The database server starts one OPT virtual processor when you use the Optical Subsystem.

Monitoring poll threads with the onstat utility

While the database server is online, you cannot drop a CPU virtual processor that is running a poll thread. To identify poll threads that run on CPU virtual processors, use the following command:

```
onstat -g ath | grep 'cpu.*poll'
```

The following **onstat -g ath** output shows two CPU virtual processors with poll threads. In this situation, you cannot drop to fewer than two CPU virtual processors.

```
tid  tcb      rstcb  prty  status   vp-class  name
8     a362b90  0      2    running  1cpu      tlitcpoll
9     a36e8e0  0      2    cond wait arrived  3cpu
```

The status field contains information, such as running, cond wait, IO Idle, IO Idle, sleeping secs: *number_of_seconds*, or sleeping forever. To improve performance, you can remove or reduce the number of threads that are identified as sleeping forever.

For more information on the types of virtual processors, see the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*.

This command has an equivalent SQL administration API function.

onmode -P: Start, stop, or restart a listen thread dynamically

Use the **onmode -P** command to start, stop, or restart an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.

Syntax:

```

▶▶ onmode -P {start|stop|restart} server_name ▶▶

```

Element	Purpose	Key Considerations
start	Start a new listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.	The definition of the listen thread must exist in the sqlhosts file for the server. If the definition of the listen thread does not exist in the sqlhosts file, you must add it before you can start the listen thread dynamically.

Element	Purpose	Key Considerations
stop	Stop an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.	The definition of the listen thread must exist in the sqlhosts file for the server.
restart	Stop and start an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.	The definition of the listen thread must exist in the sqlhosts file for the server.
server_name	The name of the database server on which you want to start, stop, or restart a listen thread.	

These commands do not update the **sqlhosts** file.

These commands are equivalent to the SQL administration API functions that have **start listen**, **stop listen**, or **restart listen** arguments.

Example

The following command stops and then starts a listen thread for a server named **ids_serv1**:

```
onmode -P restart ids_serv1
```

Related reference:

“start listen argument: Start a listen thread dynamically (SQL administration API)” on page 21-116

“stop listen argument: Stop a listen thread dynamically (SQL administration API)” on page 21-117

“restart listen argument: Stop and start a listen thread dynamically (SQL administration API)” on page 21-100

onmode -R: Regenerate .infos File

The database server uses information from the **.infos.dbservername** file when it accesses utilities. The database server creates and manages this file, and you should never need to do anything to the file. However, if **.infos.dbservername** is accidentally deleted, you must either recreate the file or shut down and restart the database server.

Syntax:

►► onmode — -R ◄◄

Element	Purpose	Key Considerations
-R	Re-creates the .infos.dbservername file	<p>Before you use the -R option, set the INFORMIXSERVER environment variable to match the DBSERVERNAME parameter from the ONCONFIG file. Do not use the -R option if INFORMIXSERVER is one of the DBSERVERALIASES names.</p> <p>For more information, see “.infos.dbservername” on page A-7.</p>

onmode -W: Change settings for the SQL statement cache

Syntax:

```
➡ onmode -W [STMT_CACHE_HITS hits | STMT_CACHE_NOLIMIT value] ➡
```

Element	Purpose	Key Considerations
STMT_CACHE_HITS <i>hits</i>	Specifies the number of hits (references) to a statement before it is fully inserted in the SQL statement cache. Set <i>hits</i> to 1 or more to exclude ad hoc queries from entering the cache.	You can only increase or reset the value of STMT_CACHE_HITS. The new value displays in the #hits field of the onstat -g ssc output. If <i>hits</i> = 0, the database server inserts all qualified statements and its memory structures in the cache. If <i>hits</i> > 0 and the number of times the SQL statement has been executed is less than STMT_CACHE_HITS, the database server inserts <i>key-only</i> entries in the cache. It inserts qualified statements in the cache after the specified number of hits have been made to the statement. ONCONFIG Parameter: STMT_CACHE_HITS
STMT_CACHE_NOLIMIT <i>value</i>	Controls whether statements are inserted in the SQL statement cache.	If <i>value</i> = 0, the database server inserts no statements in the cache. If <i>value</i> = 1, the database server always inserts statements in the cache. If none of the queries are shared, turn off STMT_CACHE_NOLIMIT to prevent the database server from allocating a large amount of memory for the cache. ONCONFIG Parameter: STMT_CACHE_NOLIMIT

Related reference:

"onmode and W arguments: Reset statement cache attributes (SQL administration API)" on page 21-85

SQL statement cache examples

The following are examples of **onmode -W** commands for changing SQL statement cache (SSC) settings. The changes are in effect for the current database server session only and do not change the ONCONFIG values. When you restart the database server, it uses the default SSC settings, if not specified in the ONCONFIG file, or the ONCONFIG settings. To make the changes permanent, set the appropriate configuration parameter.

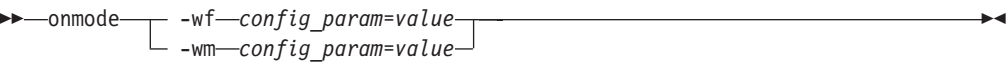
```
onmode -W STMT_CACHE_HITS 2 # number of hits before statement is
                             # inserted into SSC
onmode -W STMT_CACHE_NOLIMIT 1 # always insert statements into
                             # the cache
```

This command has an equivalent SQL administration API function.

onmode -wf, -wm: Dynamically change certain configuration parameters

Use the **onmode -wf** or **onmode -wm** command to dynamically change specific configuration parameters.

Syntax:



Element	Purpose	Key considerations
-wf	Updates the value of the specified configuration parameter in the ONCONFIG file.	None.
-wm	Dynamically sets the value of the specified configuration parameter in memory.	The specified <i>value</i> is not preserved when the server is restarted.
config_param=value	Specifies the configuration parameter and its new value.	See “Database configuration parameter reference” on page 1-24.

You can use **onmode -wm** or **onmode -wf** to change the following configuration parameters:

- ADMIN_MODE_USERS
- ALARMPROGRAM
- AUTO_AIOVPS
- AUTO_CKPTS
- AUTO_READAHEAD
- AUTO_REPREPARE
- BLOCKTIMEOUT
- CDR_DELAY_PURGE_DTC
- CDR_LOG_LAG_ACTION
- CDR_LOG_STAGING_MAXSIZE
- CKPTINTVL
- DBSPACETEMP
- DEADLOCK_TIMEOUT
- DEF_TABLE_LOCKMODE
- DELAY_APPLY
- DIRECTIVES
- DRINTERVAL
- DRTIMEOUT
- DS_MAX_QUERIES
- DS_MAX_SCANS
- DS_NONPDQ_QUERY_MEM
- DS_TOTAL_MEMORY
- DUMPCNT
- DUMPSHMEM

- DYNAMIC_LOGS
- EXPLAIN_STAT
- FILLFACTOR
- IFX_FOLDVIEW
- LIMITNUMSESSIONS
- LISTEN_TIMEOUT
- LOG_INDEX_BUILDS
- LOG_STAGING_DIR
- LOGSIZE
- LOW_MEMORY_MGR (**onmode -wf** only (not **onmode -wm**))
- LOW_MEMORY_RESERVE
- LTAPEBLK
- LTAPEDEV
- LTAPESIZE
- LTXEHWM
- LTXHWM
- MAX_INCOMPLETE_CONNECTIONS
- MAX_PDQPRIORITY
- NS_CACHE
- MSG_DATE
- MSGPATH
- NET_IO_TIMEOUT_ALARM
- ONDBSPACEDOWN
- ONLIDX_MAXMEM
- OPTCOMPIND
- RA_PAGES
- REMOTE_SERVER_CFG
- REMOTE_USERS_CFG
- RESIDENT
- RTO_SERVER_RESTART
- S6_USE_REMOTE_SERVER_CFG
- SBSPACENAME
- SBSPACETEMP
- SDS_TIMEOUT
- SHMADD
- SMX_COMPRESS
- SP_AUTOEXPAND
- SP_THRESHOLD
- SP_WAITTIME
- STACKSIZE
- STOP_APPLY
- SYSALARMPROGRAM
- SYSSBSPACENAME
- TAPEBLK
- TAPEDEV

- TAPESIZE
- TBLTBLFIRST
- TBLTBLNEXT
- TEMPTAB_NOLOG
- TXTIMEOUT
- USELASTCOMMITTED
- USTLOW_SAMPLE
- VP_MEMORY_CACHE_KB
- WSTATS

The **onmode -wf** and **onmode -wm** commands have equivalent SQL administration API functions.

Related reference:

“IFX_FOLDVIEW configuration parameter” on page 1-82

“ENABLE_SNAPSHOT_COPY Configuration Parameter” on page 1-70

“LOW_MEMORY_RESERVE configuration parameter” on page 1-90

“AUTO_READAHEAD configuration parameter” on page 1-31

“LOW_MEMORY_MGR configuration parameter” on page 1-89

“ha set ipl argument: Log index builds on the primary server (SQL administration API)” on page 21-59

“ha set timeout argument: Change SD secondary server timeout (SQL administration API)” on page 21-62

“onmode and wf arguments: Permanently update a configuration parameter (SQL administration API)” on page 21-87

“onmode and wm arguments: Temporarily update a configuration parameter (SQL administration API)” on page 21-88

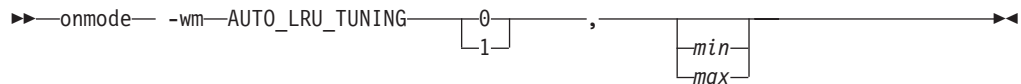
“set onconfig memory argument: Temporarily change a configuration parameter (SQL administration API)” on page 21-107

“set onconfig permanent argument: Permanently change a configuration parameter (SQL administration API)” on page 21-108

onmode -wm: Change LRU tuning status

Use the **onmode -wm** option to change the least recently used (LRU) tuning status.

Syntax:



Element	Purpose	Key considerations
-wm	Dynamically sets the value of the specified configuration parameter for the current session.	None.
0	Turns off automatic LRU tuning for the current session.	None.
1	Turns on automatic LRU tuning for the current session.	None.

Element	Purpose	Key considerations
<i>min</i>	Specifies the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory.	LRU flushing parameters can be set when LRU tuning is either on or off.
<i>max</i>	Specifies the percentage of modified pages in the LRU queues at which the queue is cleaned.	LRU flushing parameters can be set when LRU tuning is either on or off.

Important: If a *min* value is greater or equal to the *max* value for a specific buffer pool, the *min* value is ignored and no error message is returned.

This command has an equivalent SQL administration API function.

Related reference:

“onmode, wm, and AUTO_LRU_TUNING arguments: Change LRU tuning status (SQL administration API)” on page 21-89

onmode -Y: Dynamically change SET EXPLAIN

Syntax:

```

>> onmode -Y sessionid 2
                                |
                                | 1
                                |
                                | 0
  
```

Element	Purpose	Key considerations
-Y	Dynamically change the value of the SET EXPLAIN statement.	None.
<i>sessionid</i>	Identifies the specific session.	None.

You can use the SET EXPLAIN statement to display the query plan of the optimizer, an estimate of the number of rows returned, and the relative cost of the query. When you use the **onmode -Y** command to turn on SET EXPLAIN, the output is displayed in the sqexplain.out.*sessionid* file. If an sqexplain.out file already exists, the database server reads that file until an administrator turns off the SET EXPLAIN for the session.

The **onmode -Y** command dynamically changes the value of the SET EXPLAIN statement for an individual session. The following invocations are valid with this command:

Invocation	Explanation
onmode -Y sessionid 2	Turns SET EXPLAIN on for <i>sessionid</i> and displays the query plan only
onmode -Y sessionid 1	Turns SET EXPLAIN on for <i>sessionid</i>
onmode -Y sessionid 0	Turns SET EXPLAIN off for <i>sessionid</i>

For more information on using the SET EXPLAIN statement, see the *IBM Informix Guide to SQL: Syntax*. For more information on interpreting the sqexplain.out file to improve query performance, see the *IBM Informix Performance Guide*.

This command has an equivalent SQL administration API function.

Related reference:

“onmode and Y arguments: Change query plan measurements for a session (SQL administration API)” on page 21-90

onmode -z: Kill a database server session

Syntax:

►► onmode — -z *sid* ◀◀

Element	Purpose	Key considerations
-z <i>sid</i>	Kills the session that you specify in <i>sid</i>	This value must be an unsigned integer greater than 0 and must be the session identification number of a currently running session.

To use the **-z** option, first obtain the session identification (*sessid*) with **onstat -u**, then execute **onmode -z**, substituting the session identification number for *sid*.

When you use **onmode -z**, the database server attempts to kill the specified session. If the database server is successful, it frees any resources that the session holds. If the database server cannot free the resources, it does not kill the session.

If the session does not exit the section or release the latch, the database server administrator can take the database server offline, as described in “Taking the Database Server to Offline Mode with the -k Option” on page 14-16, to close all sessions.

This command has an equivalent SQL administration API function.

Related reference:

“onmode and z arguments: Terminate a user session (SQL administration API)” on page 21-91

onmode -Z: Kill a distributed transaction

Syntax:

►► onmode — -Z *address* ◀◀

Element	Purpose	Key considerations
-Z address	Kills a distributed transaction associated with the shared-memory address <i>address</i>	<p>This argument must be the address of an ongoing distributed transaction that has exceeded the amount of time that TXTIMEOUT specifies. The address must conform to the operating-system-specific rules for addressing shared-memory. (The address is available from onstat -x output.)</p> <p>This option is not valid until the amount of time that the ONCONFIG parameter TXTIMEOUT specifies has been exceeded. The -Z option should rarely be used and only by an administrator of a database server involved in distributed transactions.</p> <p>For information on initiating independent actions in a two-phase commit protocol, see the chapter on multiphase commit protocols in the <i>IBM Informix Administrator's Guide</i>.</p>

Distributed transactions provide the ability to query data on different database servers.

Attention: If applications are performing distributed transactions, killing one of the distributed transactions can leave your client/server database system in an inconsistent state. Try to avoid this situation.

This command has an equivalent SQL administration API function.

Related reference:

"onmode and Z arguments: Terminate a distributed transaction (SQL administration API)" on page 21-91

Chapter 15. The ON-Monitor Utility

Using ON-Monitor (UNIX)

Use the ON-Monitor utility to perform various administrative tasks. This section provides a quick reference for the ON-Monitor screens. To start ON-Monitor, execute the following command from the operating-system prompt:

```
onmonitor
```

If you are logged in as user **informix** or user **root**, the main menu appears. All users other than **informix** and **root** have access only to the Status menu.

The ON-Monitor main menu displays the following menus:

- **Status** menu
- **Parameters** menu
- **Dbspaces** menu
- **Mode** menu
- **Force-Ckpt** menu
- **Archive** menu
- **Logical-Logs** menu
- **Exit** option

These menus are shown on the following pages (Table 15-1 on page 15-2 through Table 15-7 on page 15-3).

To obtain ON-Monitor version information, execute the **-V** or **-version** command from the operating-system prompt. For complete information on version information, see “Obtaining Utility Version Information” on page 6-1

You cannot use ON-Monitor on High-Availability Data Replication (HDR) secondary servers, remote standalone (RS) secondary servers, or shared disk (SD) secondary servers.

Navigating ON-Monitor and Using Help

All menus and screens in ON-Monitor function in the same way. For menus, use the arrow keys or SPACEBAR to scroll to the option that you want to execute and press RETURN, or press the first capitalized letter of the option (usually the first letter). When you move from one option to the next by pressing SPACEBAR or an arrow key, the option explanation (line 2 of the menu) changes.

If you want general instructions for a specific screen, press CTRL-W. If you need help to determine what you should enter in a field on the screen, use the TAB key to highlight the field and press CTRL-F or F2.

Some of the menus display ellipses (...) on the far right or left side. The ellipses indicate that you can move in the direction of the dots, using the arrow keys or SPACEBAR, to view other options.

Executing Shell Commands Within ON-Monitor

To execute a shell command from within ON-Monitor, type an exclamation point (!) followed by the command. For example, to list the files in the current directory, type the following command:

```
!ls
```

ON-Monitor Screen Options

Table 15-1. Status Menu

Menu	Description
Profile	Displays database server performance statistics
Userthreads	Displays the status of active user threads
Spaces	Displays status information about database server storage spaces and chunks
Databases	Displays the name, owner, and logging mode of the 100 first databases
Logs	Displays status information about the physical-log buffer, the physical log, the logical-log buffer, and the logical-log files
Archive	Displays a list of all backup tapes and logical-log files that you require to restore data using ontape
data-Replication	Displays High-Availability Data-Replication (HDR) status and configuration
Output	Stores the output of other status information in a specified file
Configuration	Copies the current database server configuration to a file

Table 15-2. Parameters Menu

Menu	Description
Initialize	Initializes database server disk space or modifies disk-space parameters
Shared-Memory	Initializes database server shared memory or modifies shared-memory parameters
performance	Specifies the number of virtual processors for each VP
data-Replication	Specifies the HDR parameters
diaGnostics	Specifies values for the diagnostics parameters
pdQ	Changes parameters for parallel database queries
Add-Log	Adds a logical-log file to a dbspace
Drop-Log	Drops a logical-log file from a dbspace
Physical-Log	Changes the size or the location of the database server physical log

Table 15-3. Dbspaces Menu

Menu	Description
Create	Creates a dbspace
BLOBSpace	Creates a blobspace
Mirror	Adds mirroring to an existing storage space or ends mirroring for a storage space
Drop	Drops a storage space from the database server configuration

Table 15-3. Dbspaces Menu (continued)

Menu	Description
Info	Displays the identification number, location, and fullness of each chunk assigned to a storage space
Add_chunk	Adds a chunk to a storage space
datasKip	Changes the database parameter
Status	Changes the status of a chunk in a mirrored pair

Table 15-4. Mode Menu

Menu	Description
Startup	Initializes shared memory and takes the database server to quiescent mode
On-Line	Takes the database server from quiescent to online mode
Graceful-Shutdown	Takes the database server from online to quiescent mode so users can complete work
Immediate-Shutdown	Takes the database server from online to quiescent mode in 10 seconds
Take-Offline	Detaches shared memory and immediately takes the database server to offline mode
Add-Proc	Adds virtual processors
Drop-Proc	Drops virtual processors
Decision-support	Sets decision-support parameters dynamically
Administration	Tells the server to change into administration mode

Table 15-5. Force-Ckpt Menu

Menu	Description
Force-Ckpt	Displays the time of the most-recent checkpoint or forces the database server to execute a checkpoint

Table 15-6. Archive Menu

Menu	Description
Tape-Parameters	Modifies the ontape parameters for the backup tape device

Table 15-7. Logical Logs Menu

Menu	Description
Databases	Modifies the logging status of a database
Tape-Parameters	Modifies the ontape parameters for the logical-log backup tape device

Setting Configuration Parameters in ON-Monitor

Figure 15-1 on page 15-4 shows which ONCONFIG parameters correspond to the Initialization screen.

DISK PARAMETERS			
Page Size	[2] Kbytes		Mirror {MIRROR}
Tape Dev.	{TAPEDEV}		
Block Size	{TAPEBLK}	Total Tape Size	{TAPESIZE}
Log Tape Dev.	{LTAPEDEV}		
Block Size	{LTAPEBLK}	Total Tape Size	{LTAPESIZE}
Stage Blob	{STAGEBLOB}		
Root Name	{ROOTNAME}	Root Size	{ROOTSIZE}
Primary Path	{ROOTPATH}		
		Root Offset	{ROOTOFFSET}
Mirror Path	{MIRRORPATH}		
		Mirror Offset	{MIRROROFFSET}
Phy. Log Size	{PHYSFILE}	Log. Log Size	{LOGSIZE}
		Number of Logical Logs	{LOGFILES}

Figure 15-1. Initialization Screen with Parameter Names

Note: If you change the value of the ROOTSIZE configuration parameter after the database server is initialized, the change will not be effective until the database server is reinitialized, a process which destroys all data on the disk.

Figure 15-2 shows which ONCONFIG parameters correspond to the Shared-Memory screen.

SHARED MEMORY PARAMETERS			
Server Number	{SERVERNUM}	Server Name	{DBSERVERNAME}
Server Aliases	{DBSERVERALIASES}		
Dbpace Temp	{DBSPACETEMP}		
Deadlock Timeout	{DEADLOCK_TIMEOUT}	Dbpace Down Option	{ONDBSPACEDOWN}
Forced Residency	{RESIDENCY}	Number of Page Cleaners	{CLEANERS}
Non Res. SegSize (K)	{SHMVIRTSIZE}	Stack Size (K)	{STACKSIZE}
Heterogeneous Commit	{HETERO_COMMIT}	Optical Cache Size (K)	{OPCACHEMAX}
Physical Log Buffer Size	{PHYSBUFF}	Transaction Timeout	{TXTIMEOUT}
Logical Log Buffer Size	{LOGBUFF}	Index Page Fill Factor	{FILLFACTOR}
Max # of Locks	{LOCKS}	Add SegSize	{SHMADD}
Max # of Buffers	{BUFFERS}	Total Memory	{SHMTOTAL}
Resident Shared Memory size	[] Kbytes	Page Size	[2] Kbytes

Figure 15-2. Shared-Memory Screen with Parameter Names

Note: Although Informix can support a shared memory segment that is larger than 4 gigabytes, ON-Monitor does not support a shared memory segment that is larger than 4 gigabytes. Therefore, the ON-Monitor screen cannot hold values that are larger than 4 gigabytes.

Figure 15-3 on page 15-5 shows which ONCONFIG parameters correspond to the Performance Tuning screen.

PERFORMANCE TUNING PARAMETERS			
Multiprocessor Machine	{MULTIPROCESSOR}	LRU Max Dirty	{LRU_MAX_DIRTY}
Num Procs to Affinity	{VPCLASS aff}	LRU Min Dirty	{LRU_MIN_DIRTY}
Proc num to start with	{VPCLASS num}	Checkpoint Interval	{CKPTINTVL}
CPU VPs	{VPCLASS cpu}	Num of Read Ahead Pages	{RA_PAGES}
AIO VPs	{VPCLASS aio}	Read Ahead Threshold	{RA_THRESHOLD}
Single CPU VP	{SINGLE_CPU_VP}	NETTYPE settings:	
Use OS Time	{USE_OS_TIME}	Protocol Threads	Users VP-class
Disable Priority Aging	{VPCLASS noage}	[] [] [] []	
Offline Recovery Threads	{OFF_RECVRY_THREADS}		
Online Recovery Threads	{ON_RECVRY_THREADS}		
Num of LRU queues	{LRUS}		

Figure 15-3. Performance Screen with Parameter Names

The RA_THRESHOLD configuration parameter has been deprecated in Version 11.70. Instead you can use automatic read-ahead by setting the AUTO_READAHEAD configuration parameter.

Figure 15-4 shows which ONCONFIG parameters correspond to the Data Replication screen.

DATA REPLICATION PARAMETERS	
Interval	{DRINTERVAL}
Timeout	{DRTIMEOUT}
Lost & Found	{DRLOSTFOUND}

Figure 15-4. Data-Replication Screen with Parameter Names

Figure 15-5 shows which ONCONFIG parameters correspond to the Diagnostics screen.

DIAGNOSTIC PARAMETERS	
Message Log	{MSGPATH}
Console Msgs.	{CONSOLE}
Alarm Program	{ALARMPROGRAM}
Dump Shared Memory	{DUMPSHMEM}
Dump Gcore	{DUMPGCORE}
Dump Core	{DUMPCORE}
Dump Count	{DUMPCNT}
Dump Directory	{DUMPDIR}

Figure 15-5. Diagnostics Screen with Parameter Names

Figure 15-6 shows which ONCONFIG parameters correspond to the PDQ screen.

PARALLEL DATABASE QUERIES PARAMETERS	
Max PDQ Priority	{MAX_PDQPRIORITY}
Decision Support Queries	{DS_MAX_QUERIES}
Decision Support Memory (Kbytes)	{DS_TOTAL_MEMORY}
Maximum Decision Support Scans	{DS_MAX_SCANS}
Dataskip	{DATASKIP}
Optimizer Hint	{OPTCOMPIND}
Non PDQ Memory	{DS_NONPDQ_QUERY_MEM}

Figure 15-6. PDQ Screen with Parameter Names

```
CREATE DBSPACE
Dbospace Name [          ] Mirror [ ] Temp [ ]
Page Size [ 2] Kbytes
PRIMARY CHUNK INFORMATION:
Full Pathname [          ]
Offset [          0] Kbytes Size [          0] Kbytes
MIRROR CHUNK INFORMATION:
Full Pathname [          ]
Offset [          0] Kbytes
```

Note: All tables, indexes, and other allocations within the dbspace use pages of the specified page size. The value for Page Size must be a multiple of the page size of the root dbspace.

“AUTO_READAHEAD configuration parameter” on page 1-31

Chapter 16. The onparams Utility

Use the **onparams** utility to add or drop a logical-log file, change physical-log parameters, and add a new buffer pool.

In This Chapter

This chapter shows you how to use the following **onparams** options:

- “**onparams -a -d *dbspace***: Add a logical-log file” on page 16-2
- “**onparams -d -l *lognum***: Drop a logical-log file” on page 16-2
- “**onparams -p**: Change physical-log parameters” on page 16-3
- “**onparams -b**: Add a new buffer pool” on page 16-4

Any **onparams** command fails if a storage-space backup is in progress. If you do not use any options, **onparams** returns a usage statement.

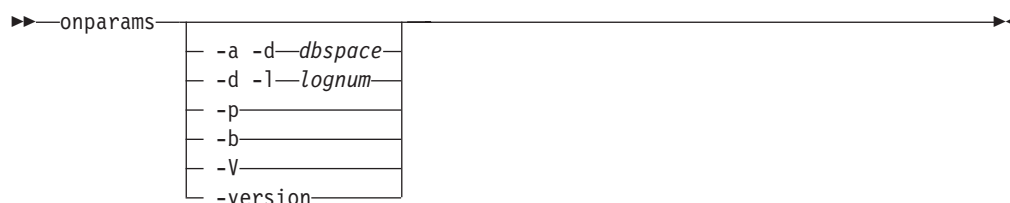
You cannot use the **onparams** utility on High-Availability Data Replication (HDR) secondary servers, remote standalone (RS) secondary servers, or shared disk (SD) secondary servers.

You can also use SQL administration API commands that are equivalent to **onparams** commands to add or drop a logical-log file, change physical-log parameters, and add a new buffer pool.

On UNIX, you must be logged in as user **root** or user **informix** to execute **onparams**. Only user **informix** is allowed to execute the SQL administration API *command* strings.

On Windows, you must be a member of the **Informix-Admin** group to execute **onparams**.

onparams Syntax



Element	Purpose	Key Considerations
-V	Displays the software version number and the serial number	See “Obtaining Utility Version Information” on page 6-1.
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See “Obtaining Utility Version Information” on page 6-1.

onparams -a -d *dbspace*: Add a logical-log file

Syntax:

►► onparams — -a — -d *dbspace* [-s *size*] [-i] ►►

Element	Purpose	Key considerations
-a -d <i>dbspace</i>	Adds a logical-log file to the end of the log-file list to the specified <i>dbspace</i>	<p>You can add a log file to a <i>dbspace</i> only if the database server has adequate contiguous space. The newly added log files have a status of A and are immediately available for use. You can add a log file during a backup. You can have a maximum of 32,767 logical-log files. Use onstat -l to view the status of your logical-log files. It is recommended that you take a level-0 backup of the root <i>dbspace</i> and the <i>dbspace</i> that contains the log file as soon as possible.</p> <p>You cannot add a log file to a <i>blob</i>space or <i>sb</i>space.</p> <p>Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i>.</p>
-i	Inserts the logical-log file after the current log file	Use this option when the Log File Required alarm prompts you to add a logical-log file.
-s <i>size</i>	Specifies a size in kilobytes for the new logical-log file	<p>This value must be an unsigned integer greater than or equal to 200 kilobytes</p> <p>If you do not specify a size with the -s option, the size of the log file is taken from the value of the LOGSIZE parameter in the ONCONFIG file when database server disk space was initialized.</p> <p>For information on changing LOGSIZE, see the chapter on managing logical-log files in the <i>IBM Informix Administrator's Guide</i>.</p>

This command has an equivalent SQL administration API function.

Related reference:

“add log argument: Add a new logical log (SQL administration API)” on page 21-16

onparams -d -l *lognum*: Drop a logical-log file

Syntax:

►► onparams — -d — -l *lognum* [-y] ►►

Element	Purpose	Key considerations
-d -l <i>lognum</i>	Allows you to drop a logical-log file specified by the log file number	<p>Restrictions: The <i>lognum</i> value must be an unsigned integer greater than or equal to 0.</p> <p>You can obtain the <i>lognum</i> from the number field of onstat -l. The sequence of <i>lognum</i> might be out of order.</p>
-y	Causes the database server to automatically respond yes to all prompts	None.

Usage

You can only drop one log files at a time.

The database server requires a minimum of three logical-log files at all times. You cannot drop a log if your logical log is composed of only three log files.

Important: Before you can drop any of the first three logical-log files, you must add new logical-log files and run a backup of the logical-log files. The backup must be run using either the **ontape -a** command or the **ontape -c** command. After you add the new logical-log files and run a backup, you can then use **onparams -d -llognum** to delete the first three logical-log files.

The status of the log file determines if the log file can be dropped, and the actions taken by the database server when the log file is dropped:

- If you drop a log file that has never been written to, status is newly Added (**A**), the database server deletes the log file and frees the space immediately.
- If you drop a used log file that has a status of User (**U**) or Free (**F**), the database server marks the log file as Deleted (**D**). After you take a level-0 backup of the dbspaces that contain the log files and the root dbspace, the database server deletes the log file and frees the space.
- You cannot drop a log file that is currently in use (**C**) or contains the last checkpoint record (**L**).

This command has an equivalent SQL administration API function.

When you move logical-log files to another dbspace, use the **onparams** commands to add and drop logical-log files. See moving a logical-log file, in the section on managing logical-log files in the *IBM Informix Administrator's Guide*.

Related reference:

“drop log argument: Drop a logical log (SQL administration API)” on page 21-47

onparams -p: Change physical-log parameters

Syntax:

►► onparams — -p — -s size — [-d dbspace] [-y] —►

Element	Purpose	Key Considerations
-p	Changes the physical log	Whenever you use the onparams -p command, you must include the -s parameter. Additionally, you can specify the -d and -y parameters. The database server must be in either the online or quiescent mode to specify the -p parameter. The database server does not need to be restarted for the changes take effect.
-s size	Changes the size (in kilobytes) of the physical log	This value must be an unsigned integer greater than or equal to 200 kilobytes. Attention: If you move the log to a dbspace without adequate contiguous space or increase the log size beyond the available contiguous space, the operation will fail and the physical log will not change.

Element	Purpose	Key Considerations
-d <i>dbspace</i>	Changes the location of the physical log to the specified <i>dbspace</i>	The space allocated for the physical log must be contiguous. Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> .
-y	Causes the database server to automatically respond yes to all prompts	None.

Related reference:

“alter plog argument: Change the physical log (SQL administration API)” on page 21-21

Backing Up After You Change the Physical-Log Size or Location

Changes to the physical log do not take effect until you restart the database server. To restart the database server immediately, execute the **onparams** command with the **-y** option.

Create a level-0 backup of the root dbspace immediately after you restart the database server. This backup is critical for proper recovery of the database server.

Changing the Size of the Physical Log and Using Non-Default Page Sizes

If you use non-default page sizes, you might need to increase the size of your physical log. If you perform many updates to non-default pages you might need a 150 to 200 percent increase of the physical log size. Some experimentation might be needed to tune the physical log. You can adjust the size of the physical log as necessary according to how frequently the filling of the physical log triggers checkpoints.

onparams -b: Add a new buffer pool

Syntax:

```

▶▶ onparams -b -g size [ -n number ] [ -r number ]
▶ [ -x percentage ] [ -m percentage ]

```

Element	Purpose	Key considerations
-b	Creates a new buffer pool	You can add a new buffer pool while the database server is running. For more information on buffer pools, see the description of the configuration parameter “BUFFERPOOL configuration parameter” on page 1-36 and the information on buffer pools in the <i>IBM Informix Administrator's Guide</i> .

Element	Purpose	Key considerations
-g <i>size</i>	Specifies the size in kilobytes of the buffer pages to create	Each dbspace you create with a non-default page size must have a corresponding buffer pool with the corresponding page size. If you create a dbspace with a page size that has no buffer pool, the system will automatically create a buffer pool using the fields in the default line of the BUFFERPOOL parameter. The size of the buffer pages must be between 2 and 16 kilobytes and it must be a multiple of the default page size.
-m <i>percent</i>	Specifies the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory	Fractional values are allowed. If you do not specify this option, the percentage used is the value of the <i>lru_min_dirty</i> field as set in the default line of the BUFFERPOOL configuration parameter. For the range of values , see “The lru_min_dirty Field” on page 1-39.
-n <i>number</i>	Specifies the number of buffers in the buffer pool	If you do not specify this option, the number used is the value of <i>buffers</i> as set in the default line of the BUFFERPOOL configuration parameter. For the range of values, see “The buffers Field” on page 1-38.
-r <i>number</i>	Specifies the number of LRU (least-recently-used) queues in the shared-memory buffer pool	If you do not include this option, the number of LRU queues allocated is equal to the value of <i>lrus</i> as set in the default line of the BUFFERPOOL configuration parameter. For the range of values, see “The lrus Field” on page 1-38.
-x <i>percent</i>	Specifies the default percentage of modified pages in the LRU queues at which the queue is cleaned	Fractional values are allowed. If you do not specify this option, the percentage used is the value of <i>lru_max_dirty</i> as set in the default line of the BUFFERPOOL configuration parameter. For the range of values , see “The lru_max_dirty Field” on page 1-39.

Create a buffer pool that corresponds to the page size of the dbspace. It is recommended that you do this before you create the dbspace. You cannot reduce or increase the number of buffers in an existing buffer pool while the database server is running. You also cannot drop a buffer pool while the database server is running. You can, however, add new buffer pools with a new size while the database server is running.

Buffer pools added with the **onparams** utility are put into virtual memory, not into resident memory. Upon restart, buffer pool entries will go into resident memory depending on the amount of memory that is available.

When you add a new buffer pool with the **onparams** utility or when you add a dbspace with a different page size (with the **onspaces** utility), the settings for the BUFFERPOOL configuration parameter in the ONCONFIG file are rewritten to reflect the new entry.

This command has an equivalent SQL administration API function.

Related reference:

“add bufferpool argument: Add a buffer pool (SQL administration API)” on page 21-14

Examples of onparams Commands

The following are examples of **onparams** commands:

```
onparams -a -d rootdbs -s 1000 # adds a 1000-KB log file to rootdbs
onparams -a -d rootdbs -i      # inserts the log file after the current log
onparams -d -l 7               # drops log 7
onparams -p -d dbpace1 -s 3000 # resizes and moves physical-log to dbpace1
onparams -b -g 6 -n 3000 -r 2 -x 2.0 -m 1.0 # adds 3000 buffers of size
6K bytes each with 2 LRUS with maximum dirty of 2% and minimum dirty of 1%
```

Chapter 17. The onpassword utility

Use the **onpassword** utility to encrypt and decrypt a password file so that the Connection Manager and Enterprise Replication utilities can connect to servers that require a secure connection.

Syntax

```
►►—onpassword— -k —access_key— —e —text_file— —d —output_filename—◄◄
```

Element	Purpose	Key Considerations
-k <i>access_key</i>	Specifies the key used to encrypt a plain text password file. The same key must be specified to decrypt the file.	The length of the <i>access_key</i> cannot exceed 24 bytes. To modify the password file, you must decrypt the file using the <i>access_key</i> that was used to encrypt the file. If the access key is lost or forgotten, decrypting the password file is not possible. In such cases, the recommended solution is to re-encrypt the plain text password file. If the plain text password file has been deleted then a new file must be created. Also see “Password key” on page 17-2.
-e <i>text_file</i>	Encrypts the specified ASCII text file that contains user IDs, passwords, and server names.	The output file is always written to \$INFORMIXDIR/etc/passwd_file. See “Password file structure” on page 17-2.
-d <i>output_filename</i>	Decrypts the specified encrypted password file.	The file is written to the specified location. See “Password file structure” on page 17-2.

Usage

The **onpassword** utility is used to encrypt or decrypt password files. A password file contains user IDs, passwords, and server information that are required for connecting to Informix servers in high-availability clusters or Enterprise Replication domains. The password file is used by the Connection Manager and the **cdr** utility. Only users logged in as **informix** have permission to run the **onpassword** utility.

You decrypt and modify the password file if you add or remove servers from your cluster or Enterprise Replication domain, change the passwords, or change your encryption key.

On both UNIX and Windows systems, an encrypted password file is required in order to configure the Connection Manager and Enterprise Replication to run in an untrusted network environment. In certain situations, an encrypted password file is also required for trusted network environments. An encrypted password file is required when a local system account attempts to connect to a server within a high-availability cluster or Enterprise Replication domain, or when the user ID does not exist on one or more servers. The password file is required because a local system account or a Windows-only account cannot directly connect to a remote server. In this case, the encrypted password file provides the correct system-level access for the request.

An encrypted password file must be located on each server on which Enterprise Replication utility commands are run. Also, an encrypted password file must be located on each server on which the Connection Manager is running.

The user IDs of the password file must have the following privileges (only user **informix** has these privileges by default):

- Permission to connect to the **sysadmin** database.
- CONNECT permission on the remote servers.
- Inclusion in the DBSA group (group **informix** on UNIX platforms, **Informix-Admin** group on Windows)

The encrypted password file cannot be copied from one platform to another. For example, password files created on a UNIX platform will not work on a Windows platform.

Password file structure

The password file consists of a separate line for each Enterprise Replication group name, each network port service, and each high-availability cluster group name. Each line also must contain the name of an alternate server, and the user ID and password for the server.

If a group is configured for both a high-availability cluster and for Enterprise Replication, there should be a line in the password file for each server in the cluster and for the group specifying the alternate server.

The input file (*text_file*) is an ASCII text file with the following structure:

```
Server_or_group AlternateServer UserName Password
```

The file contains the server names, user names, and passwords that must be specified in order to connect to the appropriate server. *AlternateServer* specifies the name of an alternate server to connect with in case the connection cannot be made to *ServerName*. For example, *AlternateServer* name is used when *ServerName* is a located on a secure port (using the s=6 option in the sqlhosts file).

Password key

The password key can be any sequence of numbers or letters up to 24 bytes. To include a space in the key, enclose the entire key in quotation marks. You must specify a key to encrypt the file and specify the same key to decrypt the file.

Examples

To encrypt a file named passwords.txt by using a key named EncryptKey and place the encrypted file in \$INFORMIXDIR/etc/passwd_file:

```
onpassword -k EncryptKey -e ./passwords.txt
```

To decrypt the \$INFORMIXDIR/etc/passwd_file and place the unencrypted file in ./passwords.txt:

```
onpassword -k EncryptKey -d ./passwords.txt
```

Configuring secure connections using a password file

You can use secure connections in a high-availability cluster, an Enterprise Replication domain, and in a combined configuration with both high-availability clusters and an Enterprise Replication domain.

High-availability cluster example

Figure 17-1 illustrates a high-availability cluster configuration. Using a secure connection in a high-availability cluster is appropriate when the servers connect over a public network. In the figure, **computer1** represents the cluster primary server and **computer2** is a secondary server.

The Connection Manager retrieves user IDs and passwords from the password file and uses them to access the primary and secondary servers in the cluster. For high-availability cluster environments, the password file must contain an entry for each group name and each alternate server name for every secondary server.

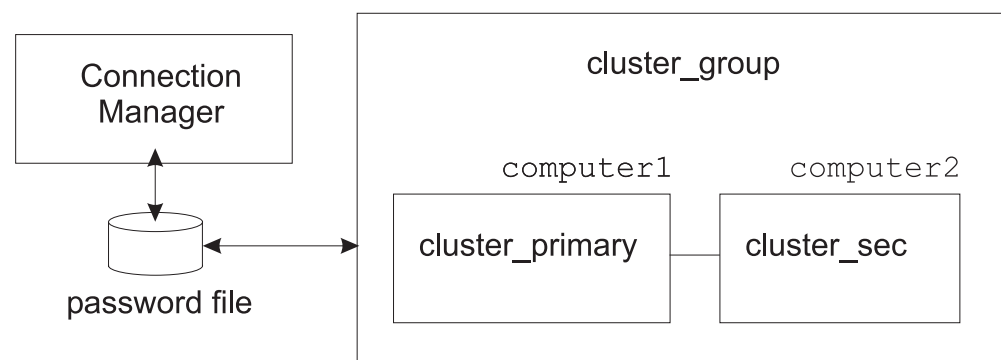


Figure 17-1. High-availability cluster password file configuration

The sqlhosts file for the high-availability cluster is configured as follows:

```
cluster_primary  ontlitcp  computer1 rach21 s=6
cluster_sec      ontlitcp  computer2 rach22 s=6

cluster_group    group     -         -
cluster_primary_a ontlitcp  computer1 rach23 g=cluster_group
cluster_sec_a    ontlitcp  computer2 rach24 g=cluster_group
```

The password file for the high-availability cluster is configured as follows:

```
cluster_group  cluster_primary_a  informix pass4DBSA
cluster_group  cluster_sec_a     informix pass4DBSA
cluster_primary cluster_primary_a  informix pass4DBSA
cluster_sec    cluster_sec_a      informix pass4DBSA
```

Enterprise Replication example

Figure 17-2 on page 17-4 illustrates a configuration with secure connections in an Enterprise Replication environment. The Enterprise Replication domain is named **er_group_1** and consists of computers named **computer1** and **computer2**.

Using a secure connection in an Enterprise Replication environment is appropriate when the servers connect over a public network.

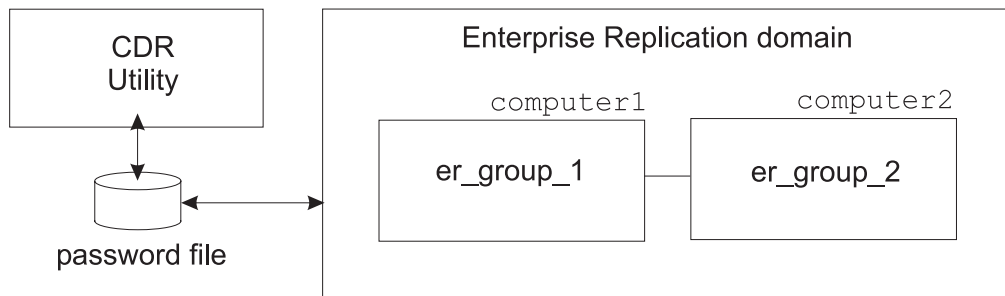


Figure 17-2. Enterprise Replication password file example

The sqlhosts file contains two entries for each computer; one entry is defined as a secure (s=6) connection and the other entry is defined without the s=6 option. The non-secure connection is used by the CDR utility, which cannot connect to a secure server. The non-secure servers are listed as alternate servers in the password file.

The password file must contain a record for each Enterprise Replication group name and alternate server name. A user ID and password must also be provided. The password file must also contain a record for each network port service server and alternate server name, and user ID and password.

The sqlhosts file for the Enterprise Replication environment is configured as follows:

```
er_group1 group - - i=100
er_srv_1 ontlitcp computer1 rach1 g=er_group1,s=6
er_srv_1a ontlitcp computer1 rach2

er_group2 group - - i=200
er_srv_2 ontlitcp computer2 rach3 g=er_group2,s=6
er_srv_2a ontlitcp computer2 rach4
```

The password file for the Enterprise Replication environment is configured as follows:

```
er_group1 er_srv_1a informix pass4DBSA
er_srv_1 er_srv_1a informix pass4DBSA
er_srv_1a er_srv_1a informix pass4DBSA

er_group2 er_srv_2a informix pass4DBSA
er_srv_2 er_srv_2a informix pass4DBSA
er_srv_2a er_srv_2a informix pass4DBSA
```

High-availability cluster with Enterprise Replication example

You configure the sqlhosts file and a password file for secure connections in an environment that contains both a high-availability cluster and Enterprise Replication. In the sqlhosts file, **computer3** is the **er_group_2** server.

The sqlhosts file for the combined high-availability cluster and Enterprise Replication environment is configured as follows:

```
# Configuration for ER ports for cluster
er_group_1 group - - i=100
cluster_primary_a ontlitcp computer1 rach1 g=er_group_1,s=6
cluster_sec_a ontlitcp computer2 rach2 g=er_group_1,s=6

# Configuration for non-ER ports for cluster
cluster_group group - -
cluster_primary ontlitcp computer1 rach3 g=cluster_group
```



```

cluster_sec      ontlitcp  computer2  rach4 g=cluster_group

# Configuration for ER ports on computer3
er_group_2       group      -          -      i=200
er_srv_2_a       ontlitcp  computer3  rach5 g=er_group_2,s=6

# Configuration for non-ER port on computer3
er_srv_2         ontlitcp  computer3  rach6

```

The password file for the combined high-availability cluster and Enterprise Replication environment is configured as follows:

```

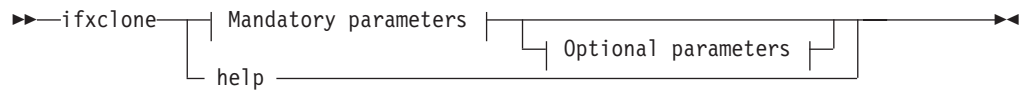
er_group_1       cluster_primary  informix  Pass4DBSA
er_group_1       cluster_sec       informix  Pass4DBSA
cluster_primary_a cluster_primary  informix  Pass4DBSA
cluster_primary  cluster_primary  informix  Pass4DBSA
cluster_sec_a    cluster_sec       informix  Pass4DBSA
cluster_group    cluster_primary  informix  pass4DBSA
cluster_group    cluster_sec       informix  pass4DBSA
er_group_2       er_srv_2          informix  Pass4DBSA
er_srv_2_a       er_srv_2          informix  Pass4DBSA
er_srv_2         er_srv_2          informix  Pass4DBSA

```

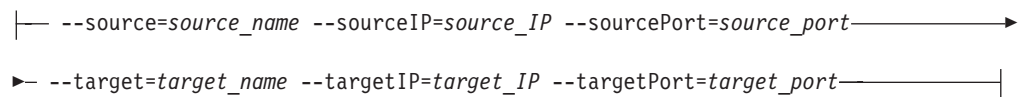

Chapter 18. The ifxclone utility

You use the **ifxclone** utility to create a snapshot of a database server.

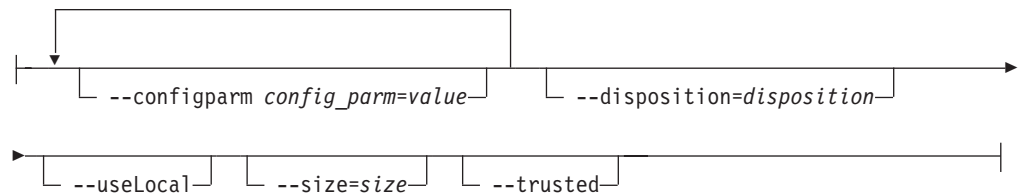
Syntax



Mandatory parameters:



Optional parameters:



Element	Purpose	Key Considerations
<i>config_parm=value</i>	Specifies an optional configuration parameter and value to set on the target server.	Certain configuration parameters on the source server must match those on the target server. See "Prerequisites for cloning an RS secondary server" on page 18-5.
<i>disposition</i>	Specifies the final disposition of the new server instance. Valid values are: RSS and ER.	If the --disposition (-d) parameter is not specified, a standard server is created.
<i>size</i>	Specifies the size of the target server. Valid values are tiny, small, medium, and large.	If the size parameter is not specified, the size parameters from the source instance are used.
<i>source_name</i>	Specifies the name of the source instance.	The source server must be a primary server and cannot be a secondary server.
<i>source_IP</i>	Specifies the source server instance TCP/IP address.	
<i>source_port</i>	Specifies the TCP/IP port address of the source server instance, or the name of a service associated with the port.	
<i>target_name</i>	Specifies the name of the target server instance.	

Element	Purpose	Key Considerations
<i>target_IP</i>	Specifies the target server instance TCP/IP address.	
<i>target_port</i>	Specifies the TCP/IP port address of the target server instance, or the name of a service associated with the port.	

The following table describes the options of the **ifxclone** utility.

Long Form	Short Form	Meaning
--configParm	-c	Specifies the name and value of a configuration parameter to set on the target server.
--disposition	-d	Specifies the disposition of the new server instance.
--help	-h	Displays usage information.
--size	-s	Specifies the size of the target instance.
--source	-S	Specifies the name of the source server instance.
--sourceIP	-I	Specifies the TCP/IP address of the source server instance.
--sourcePort	-P	Specifies the TCP/IP port address of the source server instance, or the name of a service associated with the port.
--target	-t	Specifies the name of the target server instance.
--targetIP	-i	Specifies the TCP/IP address of the target server instance.
--targetPort	-p	Specifies the TCP/IP port address of the target server instance, or the name of a service associated with the port.
--trusted	-T	Specifies that the server is trusted and that it is not necessary to obtain a userid and password to access the server.
--useLocal	-L	Specifies that the configuration information contained in the source server onconfig file should be merged with the target server onconfig file. Certain configuration parameters on the source server must match those on the target server. See "Prerequisites for all servers" on page 18-4.

Usage

Use the **ifxclone** utility to clone a server with minimum setup or configuration, or to quickly add a new node to an existing ER replication domain. When the **ifxclone** utility is run, the majority of the setup information is obtained from the server node that is being cloned. Successfully cloning a server might still require some post-configuration steps to achieve a better running system.

The source server is the server that contains the data you wish to clone. The target server is the server that is to be loaded with data from the source server. You must run the **ifxclone** utility from the target server.

To run the **ifxclone** utility on a UNIX computer, you must run the command on the target server as user root, user informix, or as a member of the informix group. You must also be a DBSA on the source server.

To run the **ifxclone** utility on a Windows computer, you must run the command on the target server as a member of the local administrators group. You must also be a DBSA on the source server and you must belong to the Informix-Admin group on the source server.

The **ifxclone** utility uses the onconfig and sqlhosts configuration files from the source server to configure the target server. The **ifxclone** utility also configures some additional configuration settings, but only those required to configure the clone server. The **ifxclone** utility is not meant to configure all of the possible configuration options, but rather to provide enough configuration to clone the source server.

The number of CPU VPs and buffers on the target server can be configured using the size parameter. Table 18-1 lists the number of CPU VPs and buffer pools created on the target server for each size option. Additional refinement of the generated configuration should be performed after the target system is created. If the size configuration is omitted, the parameter configured on the source server is used.

Table 18-1. List of size parameter values

Size	Number of CPU VPs	Number of buffers
tiny	1	50,000
small	2	100,000
medium	4	250,000
large	8	500,000

You can use the -c option to specify a configuration parameter and its value on the target server. You can also use an existing configuration file. If the target server contains a configuration file that is different than the source server configuration file, the **ifxclone** utility does not overwrite the file but modifies those parameters that must match the source server during the clone process.

The useLocal (-L) parameter is required if the target server is located on the same host machine as the source server.

If the useLocal parameter is specified, the **ifxclone** utility merges the source server onconfig file with the target server onconfig file. The configuration parameters listed in "Prerequisites for all servers" on page 18-4 are overwritten by the **ifxclone** utility and the rest of the parameters are not affected.

If the useLocal parameter is not specified as an input parameter, the **ifxclone** utility uses the source server's onconfig file as the target's onconfig file and uses the server name from the input parameters of the **ifxclone** utility.

If the `useLocal` parameter is not specified, the **ifxclone** utility updates the `sqlhosts` file on the host server with the target server entry and copies both entries to the target's `sqlhosts` file.

The order of precedence of options for the **ifxclone** parameters is as follows:

- The **--configParm** (-c) parameter takes precedence over the configuration file on the source server.
- The **--size** (-s) parameter takes precedence over merged configuration parameters or the settings in the local configuration file.
- The **--configParm** (-c) parameter takes precedence over the **--size** (-s) parameter.
- Parameters that must be the same on each server take precedence over all other options.

Prerequisites for all servers

Perform the following prerequisites before cloning a server:

- Hardware and software requirements for the servers are generally the same as those for HDR secondary servers (refer to the machine notes for specific supported platforms).
- Both the source and target servers must be part of a trusted network environment. See Network security files for information about configuring a trusted environment.
- If the disposition of the target server is specified as ER or RSS then you must provide users with connection permission to the sysadmin database on the source server. By default, connection permission to the sysadmin database is limited to user `informix`.
- Only one server clone process can occur at a time. Do not start cloning a second server until the first clone process has completed running.
- The source server must have the `ENABLE_SNAPSHOT_COPY` configuration parameter set to 1 in the `onconfig` file.

Archive operations, such as **ontape** and **ON-Bar** commands, are not allowed while cloning a server. Perform your data archive activities before starting to clone a server.

The following environment variables must be set on the target server before cloning a server:

- `INFORMIXDIR`
- `INFORMIXSERVER`
- `INFORMIXSQLHOSTS`
- `ONCONFIG`

The following configuration parameter values must be identical on both the source and target servers:

- `DRAUTO`
- `DRINTERVAL`
- `DRTIMEOUT`
- `LOGBUFF`
- `LOGFILES`
- `LOGSIZE`
- `LTAPEBLK`

- LTAPESIZE
- ROOTNAME
- ROOTSIZE
- PHYSBUFF
- PHYSFILE
- STACKSIZE
- TAPEBLK
- TAPESIZE

If the MIRROR configuration parameter is enabled on the target server, the following configuration parameters also must match between the source and target servers:

- MIRRORPATH
- MIRROROFFSET

The database server allows only certain combinations of the MIRROR configuration parameter on the source and target servers. See Table 18-2.

Table 18-2. Allowable settings of the MIRROR configuration parameter on source and target servers

MIRROR configuration parameter set on the source server	MIRROR configuration parameter set on the target server	Permitted or not permitted
No	No	Permitted
Yes	Yes	Permitted
Yes	No	Permitted
No	Yes	Not permitted. If this setting is configured, the server issues a warning and disables the MIRROR parameter in the target server onconfig file.

Prerequisites for cloning an RS secondary server

1. Set the following environment variables on the target server:
 - INFORMIXDIR
 - INFORMIXSERVER
 - ONCONFIG
 - INFORMIXSQLHOSTS
2. On the target server, create all of the chunks and mirror chunks that exist on the source server. If the target server is using mirroring, the mirror chunk paths must match those of the source server and the chunks must exist. Follow these steps to create the chunks and (if necessary) mirror chunks for the target server:
 - a. On the source server, run the **onstat -d** command to display a list of chunks and mirror chunks:


```
onstat -d
```
 - b. On the target server, log in as user informix and use the **touch**, **chown**, and **chmod** commands to create the set of chunks and mirror chunks reported by the **onstat -d** command. For example, to create a chunk named `/usr/informix/chunks/rootdbs.chunk`, follow these steps:

```

$ su informix
Password:
$ touch /usr/informix/chunks/rootdbs.chunk
$ chown informix:informix /usr/informix/chunks/rootdbs.chunk
$ chmod 660 /usr/informix/chunks/rootdbs.chunk

```

- c. Repeat all of the commands in the previous step for each chunk reported by the **onstat -d** command.
3. Run the **ifxclone** utility with the appropriate parameters on the target system.
4. Optionally, create **onconfig** and **sqlhosts** files on the target server.

Example 1, Cloning an RS secondary server using the source server configuration

This example shows how to clone a server by using the **onconfig** and **sqlhosts** configuration files from the source server.

In this example, omitting the **-L** option causes the **ifxclone** utility to retrieve the necessary configuration information from the source server. The configuration files are used as a template to create the target server configuration. Having the **ifxclone** utility create the configuration files for you saves time and reduces the chance of introducing errors into the configuration files.

For this example, assume that the source server (Amsterdam) has an **sqlhosts** file configured as follows:

```

#Server Protocol HostName Service Group
Amsterdam onsoctcp 192.168.0.1 123 -

```

You also need the name, IP address, and port number of the target server. The following values are used for this example:

- Source server name: Amsterdam
 - Source IP address: 192.168.0.1
 - Source port: 123
 - Target server name: Berlin
 - Target IP address: 192.168.0.2
 - Target port: 456
1. On the target server, log in as user **informix** and use the **touch**, **chown**, and **chmod** commands to create, change the owner, and set the permissions for the chunks. The chunk paths must match the paths of the chunks on the source server.

2. On the target server, run the **ifxclone** utility:

```

ifxclone -T -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
-i 192.168.0.2 -p 456 -d RSS

```

The **ifxclone** utility modifies the **sqlhosts** file on the source server and creates a copy of the file on the new target server. The **sqlhosts** file on the target server is the same as the source server:

```

#Server Protocol HostName Service Group
Amsterdam onsoctcp 192.168.0.1 123 -
Berlin onsoctcp 192.168.0.2 456

```

Example 2, Cloning an RS secondary server by merging the source server configuration

Use the **-L** (**--useLocal**) option to create a clone of a server on a remote host computer: The **-L** option is used to merge the source **onconfig** file configuration

information with the target onconfig file. This option also copies the source sqlhosts file to the target server. The following values are used for this example:

- Source server name: Amsterdam
 - Source IP address: 192.168.0.1
 - Source port: 123
 - Target server name: Berlin
 - Target IP address: 192.168.0.2
 - Target port: 456
1. Create the onconfig and sqlhosts files and set the environment variables on the target computer.
 2. On the target server, log in as user informix and use the **touch**, **chown**, and **chmod** commands to create, change the owner, and set the permissions for the chunks. The chunk paths must match the paths of the chunks on the source server.
 3. On the target server, run the **ifxclone** utility:

```
ifxclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
-i 192.168.0.2 -p 456 -d RSS
```

Example 3, Adding an RS secondary server to a cluster

This example shows how to add an RS secondary server to the existing Informix high-availability cluster. The following values are used for this example:

- Source server name: Amsterdam
 - Source IP address: 192.168.0.1
 - Source port: 123
 - Target server name: Berlin
 - Target IP address: 192.168.0.2
 - Target port: 456
1. Create the onconfig and sqlhosts files and set the environment variables on the target computer.
 2. On the target server, log in as user informix and use the **touch**, **chown**, and **chmod** commands to create, change the owner, and set the permissions for the chunks. The chunk paths must match the paths of the chunks on the source server.
 3. On the target server, run the **ifxclone** utility:

```
ifxclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
-i 192.168.0.2 -p 456 -s medium -d RSS
```

Prerequisites for cloning an ER server

Complete the following prerequisites before attempting to clone an ER server.

1. The source server (that is, the server that is being cloned) must have ER configured and active.
2. For configuration parameters that specify directory names, the directory names must exist on the target server. For example, if the CDR_LOG_STAGING_DIR configuration parameter is set to a directory name on the source server then the directory must also exist on the target server.
3. If ATS or RIS is enabled on the source server then the appropriate ATS or RIS directories must exist on the target server. See Enabling ATS and RIS File Generation and Creating ATS and RIS Directories. If the directories do not exist then ATS/RIS spooling will fail.

4. If the source server has the CDR_SERIAL configuration parameter set then you must set the value for CDR_SERIAL to a different value on the server to be cloned. The value of CDR_SERIAL must be different on all replication servers. You can specify a unique value for the CDR_SERIAL configuration parameter by using the **--configParm (-c)** parameter in the **ifxclone** command line.
5. The clock on the new ER clone must be appropriately synchronized. See Time Synchronization.
6. The source server (that is, the server being cloned) must not have any stopped or suspended replicates, nor can it have any shadow replicates defined.

Avoid performing ER administrative tasks that change the set of replicates on which the target server participates while the **ifxclone** utility is running.

Example: Creating a clone of an ER server

Suppose you have five ER servers named S1, S2, S3, S4, and S5 currently configured as root servers in an ER domain. You would like to add a new server, S6, on a new computer named machine6, and you want it to have the same data as server S3.

1. Install and configure Informix database software on machine6. You can use the deployment utility to deploy a pre-configured database server instance.
2. Copy the `sqlhosts` file from server S3 to server S6 and modify it to add entries for the new server. For example, assuming the ER group name for the new server is `g_S6` and the ID is 60, the `sqlhosts` file lines would look like the following.

```
g_S6      group      -          -          i=60
S6        onsoctcp   machine6    service6  g=g_S6
```
3. Add the two lines from the previous step in the `sqlhosts` files on all of the other five servers (S1 through S5).
4. Copy the `onconfig` file from server S3 to server S6 and change the `DBSERVERNAME` configuration parameter to S6. Do not modify any storage or chunk parameters except for path information.
5. On server S6 (machine6) provision chunk paths and other storage to the same sizes as server S3. Ensure that S6 has adequate memory and disk space resources.
6. Run the following command as user **informix**:

```
ifxclone -L -S S3 -I machine3 -P service3 -t S6 -i machine6 -p service6 -d ER
```

When prompted, enter the user name **informix** and then enter the password for user **informix**.
7. Monitor the server logs of servers S6 and S3. When the cloning process is complete you can check the status of servers by running the following command on servers S3 and S6:

```
cdr list server
```

You should see the new ER server `g_S6` connected to all of the other five servers. In addition, ER node `g_S6` will now participate in all replicates in which ER node `g_S3` participates.

Related reference:

“ENABLE_SNAPSHOT_COPY Configuration Parameter” on page 1-70

Chapter 19. The onspaces Utility

This chapter shows you how to use the following **onspaces** options:

- “onspaces -a: Add a chunk to a dbspace or blobspace” on page 19-2
- “onspaces -a: Add a chunk to an sbspace” on page 19-3
- “onspaces -c -b: Create a blobspace” on page 19-5
- “onspaces -c -d: Create a dbspace” on page 19-7
- “onspaces -c -S: Create an sbspace” on page 19-11
- “onspaces -c -x: Create an extspace” on page 19-16
- “onspaces -ch: Change sbspace default specifications” on page 19-17
- “onspaces -cl: Clean up stray smart large objects in sbspaces” on page 19-18
- “onspaces -d: Drop a chunk in a dbspace, blobspace, or sbspace” on page 19-18
- “onspaces -d: Drop a blobspace, dbspace, extspace, or sbspace” on page 19-20
- “onspaces -f: Specify DATASKIP parameter” on page 19-21
- “onspaces -m: Start mirroring” on page 19-22
- “onspaces -r: Stop mirroring” on page 19-24
- “onspaces -ren: Rename a dbspace, blobspace, sbspace, or extspace” on page 19-24
- “onspaces -s: Change status of a mirrored chunk” on page 19-26

When you use **onspaces** or ISA to manage a storage space, the database server updates information about the space in the **oncfg_servername.servernum** file. For more information on the **oncfg*** file, refer to Appendix A, “Files That the Database Server Uses,” on page A-1.

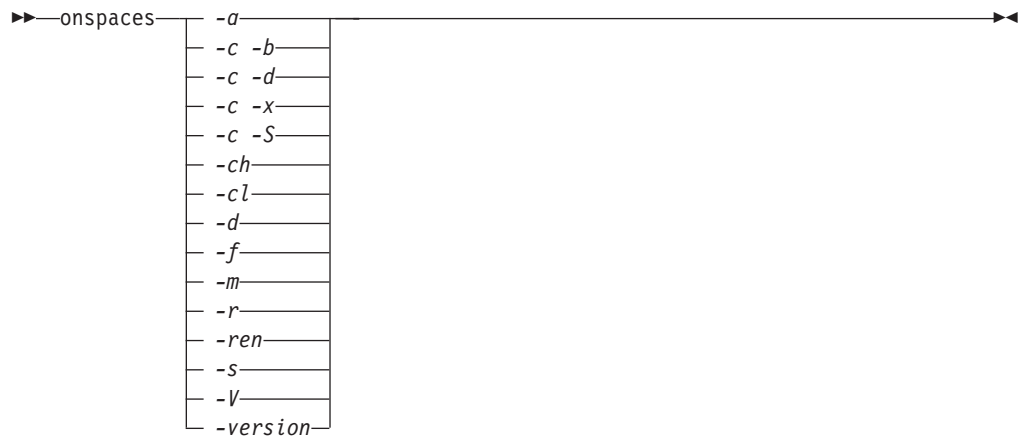
You can specify a maximum of 2047 chunks for a storage space, and a maximum of 2047 storage spaces on the database server system. The storage spaces can be any combination of dbspaces, blobspaces, and sbspaces.

On UNIX, you must be logged in as user **root** or user **informix** to execute **onspaces**. On Windows, you must be a member of the **Informix-Admin** group.

You cannot use the **onspaces** utility on High-Availability Data Replication (HDR) secondary servers, remote standalone (RS) secondary servers, or shared disk (SD) secondary servers.

You can use SQL administration API commands that are equivalent to **onspaces** commands.

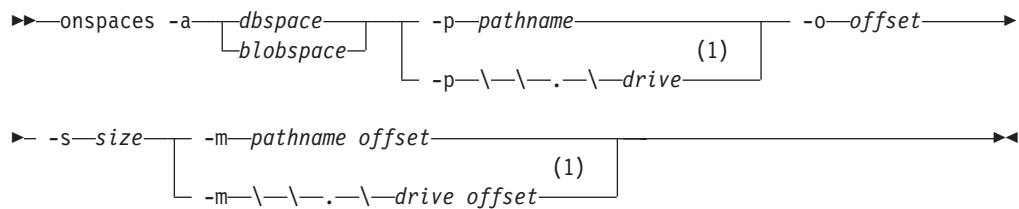
onspaces Syntax



Element	Purpose	Key Considerations
-V	Displays the software version number and the serial number	See "Obtaining Utility Version Information" on page 6-1
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See "Obtaining Utility Version Information" on page 6-1

onspaces -a: Add a chunk to a dbspace or blobspace

Syntax:



Notes:

- 1 Windows only

Use **onspaces -a** to add a chunk to a dbspace or blobspace.

Element	Purpose	Key considerations
-a	Indicates that a chunk is to be added	You can have up to 32766 chunks in an instance. You can put all those chunks in one storage space, or spread them among multiple storage spaces.
drive	Specifies the Windows drive to allocate as unbuffered disk space. The format can be either <code>\\.\<drive></code> , where <i>drive</i> is the drive letter assigned to a disk partition, or <code>\\.\PhysicalDrive<number></code> , where <i>PhysicalDrive</i> is a constant value and <i>number</i> is the physical drive number.	For more information on allocating unbuffered disk space, see in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> . Example: <code>\\.\F:</code> For path name syntax, see your operating-system documentation.

Element	Purpose	Key considerations
-m pathname offset	Specifies an optional path name and offset to the chunk that mirrors the new chunk. Also see the entries for <i>pathname</i> and <i>offset</i> in this table.	For more information, see adding a chunk to a dbspace or blobspace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-o offset	After the -a option, <i>offset</i> indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace or dbspace.	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes. For more information, see allocating raw disk space on UNIX in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-p pathname	Indicates the disk partition or unbuffered device of the initial chunk of the blobspace or dbspace that you are adding. The chunk must be an existing unbuffered device or buffered file.	The chunk name can be up to 128 bytes. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. UNIX example (unbuffered device): /dev/rdisk/c0t3d0s4 UNIX example (buffered device): /ix/ids9.2/db1chunk Windows example: c:\Ifmxdta\ol_icecream\mychunk1.dat For path name syntax, see your operating-system documentation.
-s size	Indicates, in kilobytes, the size of the new blobspace or dbspace chunk.	Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes.
blobspace	Names the blobspace to which you are adding a chunk.	See adding a chunk to a blobspace in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> . Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> .
dbspace	Names the dbspace to which you are adding a chunk.	See adding a chunk to a dbspace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> . Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> .

This command has an equivalent SQL administration API function.

Related reference:

“Avoid overwriting a chunk” on page 19-27

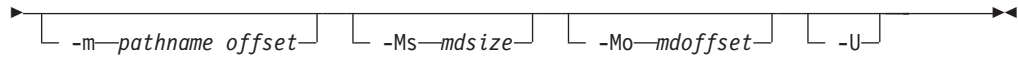
“add chunk argument: Add a new chunk (SQL administration API)” on page 21-15

“create chunk argument: Create a chunk (SQL administration API)” on page 21-29

onspaces -a: Add a chunk to an sbpace

Syntax:

►► onspaces -a—*sbspace*— -p—*pathname*— -o—*offset*— -s—*size*—►►



Use **onspaces -a** to add a chunk to an sbospace.

Element	Purpose	Key considerations
-a	Indicates that a chunk is to be added	You can have up to 32766 chunks in an instance. You can put all those chunks in one storage space, or spread them among multiple storage spaces.
-m <i>pathname offset</i>	Specifies an optional path name and offset to the chunk that mirrors the new chunk. Also see the entries for <i>pathname</i> and <i>offset</i> in this table.	For background information, see adding a chunk to an sbospace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-Mo <i>mdoffset</i>	Indicates, in kilobytes, the offset into the disk partition or into the device where metadata should be stored	Value can be an integer between 0 and the chunk size. You cannot specify an offset that causes the end of the metadata space to be past the end of the chunk. For background information, see sizing sbospace metadata, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-Ms <i>mdsize</i>	Specifies the size, in kilobytes, of the metadata area allocated in the initial chunk. The remainder is user-data space	Value can be an integer between 0 and the chunk size. For background information, see sizing sbospace metadata, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-o <i>offset</i>	After the -a option, <i>offset</i> indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the new blob space or db space.	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 terabytes, depending on the platform. For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-p <i>pathname</i>	Indicates the disk partition or unbuffered device of the initial chunk of the sbospace that you are creating The chunk must be an existing unbuffered device or buffered file.	The chunk name can be up to 128 bytes. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. For path name syntax, see your operating-system documentation.
-U	Specifies that the entire chunk should be used to store user data	The -M and -U options are mutually exclusive. For background information, see adding a chunk to an sbospace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-s <i>size</i>	Indicates, in kilobytes, the size of the new sbospace chunk	Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes

Element	Purpose	Key considerations
<i>sbspace</i>	Names the sbspace to which you are adding a chunk	See adding a chunk to an sbspace in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> . Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> .

This command has an equivalent SQL administration API function.

Related reference:

“Avoid overwriting a chunk” on page 19-27

“add chunk argument: Add a new chunk (SQL administration API)” on page 21-15

“create chunk argument: Create a chunk (SQL administration API)” on page 21-29

onspaces -c -b: Create a blobspace

Syntax:

```

▶▶ onspaces -c -b blobspace -g pageunit -p pathname -o offset
                                   (1)
                                   -p \- \- . - \- drive
▶ -s size
    -m pathname offset
    (1)
    -m \- \- . - \- drive offset

```

Notes:

- 1 Windows Only

Use **onspaces -c -b** to create a blobspace.

Element	Purpose	Key considerations
-b <i>blobspace</i>	Names the blobspace to be created	The blobspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character. For more information, see creating a blobspace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> . The syntax must conform to the Identifier segment. For more information, see the <i>IBM Informix Guide to SQL: Syntax</i> .
-c	Creates a dbspace, blobspace, sbspace, or extspace You can create up to 2047 storage spaces of any type.	After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. For more information, see creating a dbspace, blobspace, or extspace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .

Element	Purpose	Key considerations
<i>drive</i>	Specifies the Windows drive to allocate as unbuffered disk space The format can be either <code>\\.\<drive></code> , where <i>drive</i> is the drive letter assigned to a disk partition, or <code>\\.\PhysicalDrive<number></code> , where <i>PhysicalDrive</i> is a constant value and <i>number</i> is the physical drive number.	For information on allocating unbuffered disk space, see allocating unbuffered disk space on Windows in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> . Examples: <code>\\.\F:</code> <code>\\.\PhysicalDrive2</code> For path name syntax, see your operating-system documentation.
-g <i>pageunit</i>	Specifies the blobspace blobpage size in terms of <i>page_unit</i> , the number of disk pages per blobpage	Unsigned integer. Value must be greater than 0. The maximum number of pages that a blobspace can contain is 2147483647. Therefore, the size of the blobspace is limited to the blobpage size x 2147483647. This includes blobpages in all chunks that make up the blobspace. For more information, see blobpage size considerations, in the chapter on I/O Activity in the <i>IBM Informix Performance Guide</i> .
-m <i>pathname offset</i>	Specifies an optional path name and offset to the chunk that mirrors the initial chunk of the new blobspace or dbspace Also see the entries for -p <i>pathname</i> and -o <i>offset</i> in this table.	For more information, see creating a dbspace or a blobspace in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-o <i>offset</i>	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace, dbspace, or sbpace	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 terabytes, depending on the platform. For more information, see allocating raw disk space, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-p <i>pathname</i>	Indicates the disk partition or device of the initial chunk of the blobspace or dbspace that you are creating	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. UNIX example (unbuffered device): <code>/dev/rdisk/c0t3d0s4</code> UNIX example (buffered device): <code>/ix/ids9.2/db1chunk</code> Windows example: <code>c:\Ifmxdata\ol_icecream\mychunk1.dat</code> For path name syntax, see your operating-system documentation.
-s <i>size</i>	Indicates, in kilobytes, the size of the initial chunk of the new blobspace or dbspace	Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.

This command has an equivalent SQL administration API function.

Related reference:

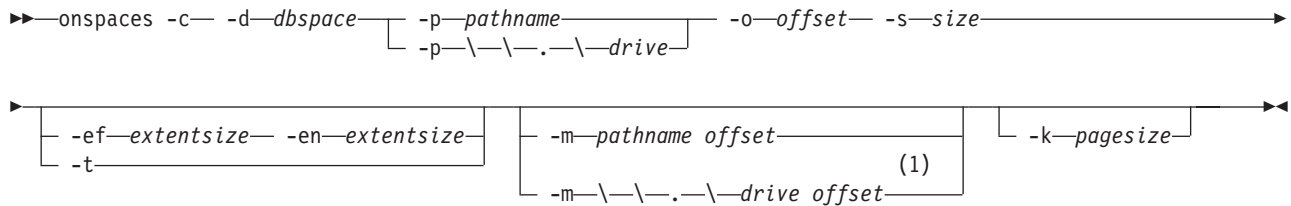
“Avoid overwriting a chunk” on page 19-27

“create blobspace argument: Create a blobspace (SQL administration API)” on page 21-26

onspaces -c -d: Create a dbspace

Use the **onspaces -c -d** option to create a dbspace or a temporary dbspace.

Syntax:



Notes:

1 Windows Only

Element	Purpose	Key considerations
-c	Creates a dbspace You can create up to 2047 storage spaces of any type.	After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. For more information, see Manage dbspaces (Administrator's Guide).
<i>drive</i>	Specifies the Windows drive to allocate as unbuffered disk space The format can be either <code>\\.\<drive></code> , where <i>drive</i> is the drive letter assigned to a disk partition, or <code>\\.\PhysicalDrive<number></code> , where <i>PhysicalDrive</i> is a constant value and <i>number</i> is the physical drive number.	For information on allocating unbuffered disk space, see Allocating raw disk space on Windows (Administrator's Guide). Examples: <code>\\.\F:</code> <code>\\.\PhysicalDrive2</code> okay For path name syntax, see your operating-system documentation.
-d dbspace	Names the dbspace to be created	The dbspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character. For more information, see Manage dbspaces (Administrator's Guide). The syntax must conform to the Identifier segment. For more information, see the <i>IBM Informix Guide to SQL: Syntax</i> .

Element	Purpose	Key considerations
-ef extentsize	Indicates, in kilobytes, the size of the first extent for the tblspace	<p>The minimum, and default, size of the first extent for the tblspace of a non-root dbspace is equivalent to 50 dbspace pages, specified in K. For example: 100 KB for a 2 KB page size dbspace, 200 KB for a 4 KB page size dbspace, 400 KB for an 8 KB page size dbspace.</p> <p>The maximum size of a tblspace extent is 1048575 pages minus the space needed for any system objects. On a 2 KB pagesize system this would evaluate to approximately 2 GB.</p> <p>For more information, see Specifying the first and next extent sizes for the tblspace (Administrator's Guide).</p>
-en extentsize	Indicates, in kilobytes, the size of the next extents in the tblspace	<p>The minimum size of the next extents for the tblspace of a non-root dbspace is equivalent to 4 dbspace pages, specified in K. For example: 8 KB for a 2 KB page size dbspace, 16 KB for a 4 KB page size dbspace, 32 KB for an 8 KB page size dbspace.</p> <p>The default size for a next extent is 50 dbspace pages.</p> <p>The maximum size of a tblspace extent is 1048572 pages. On a 2 KB pagesize system this would evaluate to approximately 2 GB.</p> <p>If there is not enough space for a next extent in the primary chunk, the extent is allocated from another chunk. If the specified space is not available, the closest available space is allocated.</p> <p>For more information, see Specifying the first and next extent sizes for the tblspace (Administrator's Guide).</p>
-k pagesize	<p>Indicates in kilobytes, the non-default page size for the new dbspace.</p> <p>For systems with sufficient storage, performance advantages of a larger page size can include the following:</p> <ul style="list-style-type: none"> • Reduced depth of B-tree indexes, even for smaller index keys • You can group on the same page long rows that currently span multiple pages of the default page size • Checkpoint time is typically reduced with larger pages • You can define a different page size for temporary tables, so that they have a separate buffer pool. 	<p>The page size must be between 2KB and 16KB and must be a multiple of the default page size. For example, if the default page size is 2KB, then <i>pagesize</i> can be 2, 4, 6, 8, 10, 12, 14, or 16. If the default page size is 4KB (Windows), then <i>pagesize</i> can be 4, 8, 12, or 16.</p> <p>For more information, see Creating a dbspace with a non-default page size (Administrator's Guide).</p>

Element	Purpose	Key considerations
-m <i>pathname offset</i>	Specifies an optional path name and offset to the chunk that mirrors the initial chunk of the new dbspace Also see the entries for -p <i>pathname</i> and -o <i>offset</i> in this table.	For more information, see Manage dbspaces (Administrator's Guide).
-o <i>offset</i>	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new dbspace	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The offset must be a multiple of the page size. The maximum offset is 2 or 4 terabytes, depending on the platform. For more information, see Allocating raw disk space on Windows (Administrator's Guide).
-p <i>pathname</i>	Indicates the disk partition or device of the initial chunk of the dbspace that you are creating	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. UNIX example (unbuffered device): /dev/rdisk/c0t3d0s4 UNIX example (buffered device): /ix/ids9.2/db1chunk Windows example: c:\Ifmxdata\ol_icecream\mychunk1.dat For path name syntax, see your operating-system documentation.
-s <i>size</i>	Indicates, in kilobytes, the size of the initial chunk of the new dbspace	Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.
-t	Creates a temporary dbspace for storage of temporary tables	You cannot mirror a temporary dbspace. You cannot specify the first and next extent sizes for the tblspace of a temporary dbspace. For more information, see Temporary dbspaces (Administrator's Guide).

The maximum size of a dbspace is equal to the maximum number of chunks multiplied by the maximum size of a chunk. (The maximum number of chunks is 32766 per instance. The maximum size of a chunk is equal to 2147483647 pages multiplied by the page size.)

This command has an equivalent SQL administration API function.

Related reference:

“Avoid overwriting a chunk” on page 19-27

“create dbspace argument: Create a dbspace (SQL administration API)” on page 21-32

“create tempdbspace argument: Create a temporary dbspace (SQL administration API)” on page 21-38

Creating a Temporary Dbspace with the -t Option

When you create a temporary dbspace with **onspaces**, the database server uses the newly created temporary dbspace, after you perform the following steps:

1. Add the name of the new temporary dbspace to your list of temporary dbspaces in the DBSPACETEMP configuration parameter, the DBSPACETEMP environment variable, or both.
2. Restart the database server.

Use **create tempdbspace** as the SQL administration API *command* string for **onspaces -c -d -t**.

Specifying First and Next Extent Size for the tblspace tblspace

You cannot specify the first and next extent of a temporary dbspace. The extent size for temporary dbspaces is 100 kilobytes for a 2 kilobyte page system or 200 kilobytes for a 4 kilobyte page system.

To specify the first and next extent sizes of a root tblspace **tblspace**, use the TBLTBLFIRST and TBLTBLNEXT configuration parameters before you create the root dbspace the first time you start the database server.

Specifying a Non-Default Page Size with the Same Size as the Buffer Pool

When you create a dbspace with a non-default page size, you must also create a buffer pool specific to that page size. It is recommended that you create the buffer pool before you create the dbspace. Use the **onparams** utility to create a buffer pool. For more information, see “onparams -b: Add a new buffer pool” on page 16-4.

When you add a dbspace with a different page size with the **onspaces** utility or you add a new buffer pool (with the **onparams** utility), a new BUFFERPOOL line is appended in the BUFFERPOOL configuration parameter in the ONCONFIG file to reflect the new entry and it is rewritten to disk.

Notes:

1. You cannot change the page size of a dbspace after you create it.
2. You cannot store logical or physical logs in a dbspace that is not the default platform page size.
3. If a dbspace is created when a buffer pool with that page size does not exist, Informix creates a buffer pool using the values of the fields of the default line of the BUFFERPOOL parameter. You cannot have multiple buffer pools with the same page size.

onspaces -c -S: Create an sbpace

Use the **onspaces -c -S** option to create a sbpace or a temporary sbpace.

Syntax:

```
➤ onspaces -c -S sbpace [-t] -p pathname -o offset -s size ➤
➤ [-m pathname offset] [-Ms mdspace] [-Mo mdoffset] ➤
➤ [-Df default list] ➤
```

Element	Purpose	Key Considerations
-S sbpace	Names the sbpace to be created	The sbpace name must be unique and must not exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character. Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> .
-c	Creates an sbpace You can create up to 32767 storage spaces of any type.	None.
-m pathname offset	Specifies an optional pathname and offset to the chunk that mirrors the initial chunk of the new sbpace Also see the entries for -p pathname and -o offset in this table.	For more information, see sbspaces in the chapter on data storage, and creating an sbpace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-Mo mdoffset	Indicates, in kilobytes, the offset into the disk partition or into the device where metadata will be stored.	Restrictions: Value can be an integer between 0 and the chunk size. You cannot specify an offset that causes the end of the metadata space to be past the end of the chunk. References: For more information, see sizing sbpace metadata, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-Ms mdspace	Specifies the size, in kilobytes, of the metadata area allocated in the initial chunk The remainder is user-data space.	Restrictions: Value can be an integer between 0 and the chunk size.
-o offset	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the sbpace	Restrictions: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 4 terabytes for systems with a two-kilobyte page size and 8 terabytes for systems with a four-kilobyte page size. References: For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .

Element	Purpose	Key Considerations
-p <i>pathname</i>	Indicates the disk partition or unbuffered device of the initial chunk of the sbpace	<p>The chunk must be an existing unbuffered device or buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server.</p> <p>References: For pathname syntax, see your operating-system documentation.</p>
-s <i>size</i>	Indicates, in kilobytes, the size of the initial chunk of the new sbpace	<p>Restrictions: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size.</p> <p>The maximum chunk size is 2 or 4 terabytes, depending on the platform.</p>
-t	Creates a temporary sbpace for storage of temporary smart large objects. You can specify the size and offset of the metadata area	<p>Restrictions: You cannot mirror a temporary sbpace. You can specify any -Df option, except the LOGGING=ON option, which has no effect.</p> <p>References: For more information, see “Creating a Temporary Sbpac with the -t Option.”</p>
-Df <i>default list</i>	Lists default specifications for smart large objects stored in the sbpace	<p>Restrictions: Tags are separated by commas. If a tag is not present, system defaults take precedence. The list must be enclosed in double quotation marks (") on the command line.</p> <p>References: For a list of tags and their parameters, see Table 19-1 on page 19-13.</p>

Related reference:

“Avoid overwriting a chunk” on page 19-27

“create sbpace argument: Create an sbpace (SQL administration API)” on page 21-34

“create tempsbpac argument: Create a temporary sbpace (SQL administration API)” on page 21-39

“create sbpace with log argument: Create an sbpace with transaction logging (SQL administration API)” on page 21-37

Creating a Temporary Sbpac with the -t Option

This example creates a temporary sbpace of 1000 kilobytes:

```
onspaces -c -S tempsbp -t -p ./tempsbp -o 0 -s 1000
```

You can optionally specify the name of the temporary sbpace in the SBSPACETEMP configuration parameter. Restart the database server so that it can use the temporary sbpace.

Creating an Sbpac with the -Df option

When you create an sbpace with the optional **-Df** option, you can specify several default specifications that affect the behavior of the smart large objects stored in the sbpace. The default specifications must be expressed as a list separated by

commas. The list need not contain all of the tags. The list of tags must be enclosed in double quotation marks ("). The table in Table 19-1 describes the tags and their default values.

The four levels of inheritance for sbspace characteristics are system, sbspace, column, and smart large objects. For more information, see smart large objects in the chapter on where data is stored in the *IBM Informix Administrator's Guide*.

Table 19-1. -Df Default Specifications

Tag	Values	Default	Description
ACCESSTIME	ON or OFF	OFF	<p>When set to ON, the database server tracks the time of access to all smart large objects stored in the sbspace.</p> <p>For information about altering storage characteristics of smart large objects, see the <i>IBM Informix DataBlade API Programmer's Guide</i>.</p>
AVG_LO_SIZE	<p>Windows: 4 to 2**31</p> <p>UNIX: 2 to 2**31</p>	8	<p>Specifies the average size, in kilobytes, of the smart large object stored in the sbspace</p> <p>The database server uses this value to calculate the size of the metadata area. Do not specify AVG_LO_SIZE and -Ms together. You can specify AVG_LO_SIZE and the metadata offset (-Mo) together.</p> <p>If the size of the smart large object exceeds 2**31, specify 2**31. If the size of the smart large object is less than 2 on UNIX or less than 4 in Windows, specify 2 or 4.</p> <p>Error 131 is returned if you run out of space in the metadata and reserved areas in the sbspace. To allocate additional chunks to the sbspace that consist of metadata area only, use the -Ms option instead.</p> <p>For more information, see creating smart large objects, in the chapter on managing data on disk in the <i>IBM Informix Administrator's Guide</i>.</p>
BUFFERING	ON or OFF	ON	<p>Specifies the buffering mode of smart large objects stored in the sbspace</p> <p>If set to ON, the database server uses the buffer pool in the resident portion of shared memory for smart-large-object I/O operations. If set to OFF, the database server uses light I/O buffers in the virtual portion of shared memory (lightweight I/O operations).</p> <p>BUFFERING = OFF is incompatible with LOCK_MODE = RANGE and creates a conflict</p> <p>For more information, see lightweight I/O, in the chapter on configuration effects on memory in the <i>IBM Informix Performance Guide</i>.</p>

Table 19-1. -Df Default Specifications (continued)

Tag	Values	Default	Description
LOCK_MODE	RANGE or BLOB	BLOB	<p>Specifies the locking mode of smart large objects stored in the sbspace</p> <p>If set to RANGE, only a range of bytes in the smart large object is locked. If set to BLOB, the entire smart large object is locked.</p> <p>LOCK_MODE = RANGE is incompatible with BUFFERING = OFF and creates a conflict.</p> <p>For more information, see smart large objects, in the chapter on locking in the <i>IBM Informix Performance Guide</i>.</p>
LOGGING	ON or OFF	OFF	<p>Specifies the logging status of smart large objects stored in the sbspace</p> <p>If set to ON, the database server logs changes to the user data area of the sbspace. When you turn on logging for an sbspace, take a level-0 backup of the sbspace.</p> <p>When you turn off logging, the following message displays: You are turning off smart large object logging.</p> <p>For more information, see smart large objects, in the chapters on data storage and logging in the <i>IBM Informix Administrator's Guide</i>. For information about onspaces -ch messages, see Appendix E, "Error Messages," on page E-1.</p>
EXTENT_SIZE	4 to 2**31	None	<p>Specifies the size, in kilobytes, of the first allocation of disk space for smart large objects stored in the sbspace when you create the table</p> <p>Let the system select the EXTENT_SIZE value. To reduce the number of extents in a smart large object, use mi_lo_specset_estbytes (DataBlade API) or ifx_lo_specset_estbytes (Informix ESQ/C) to hint to the system the total size of the smart large object. The system attempts to allocate a single extent for the smart large object.</p> <p>For more information, see smart large objects, in the chapter on where data is stored in the <i>IBM Informix Administrator's Guide</i>. For information about altering storage characteristics of smart large objects, see the <i>IBM Informix DataBlade API Programmer's Guide</i> or the <i>IBM Informix ESQ/C Programmer's Manual</i>.</p>
MIN_EXT_SIZE	2 to 2**31	Windows: 4 UNIX: 2	<p>Specifies the minimum amount of space, in kilobytes, to allocate for each smart large object</p> <p>The following message displays: Changing the sbspace minimum extent size: old value <i>value1</i> new value <i>value2</i>.</p> <p>For information about tuning this value, see smart large objects, in the chapter on configuration effects on I/O utilization in the <i>IBM Informix Performance Guide</i>. For information about onspaces -ch messages, see Appendix E, "Error Messages," on page E-1.</p>

Table 19-1. -Df Default Specifications (continued)

Tag	Values	Default	Description
NEXT_SIZE	4 to 2**31	None	<p>Specifies the extent size, in kilobytes, of the next allocation of disk space for smart large objects when the initial extent in the sbspace becomes full. Let the system select the NEXT_SIZE value. To reduce the number of extents in a smart large object, use mi_lo_specset_estbytes or ifx_lo_specset_estbytes to hint to the system the total size of the smart large object. The system attempts to allocate a single extent for the smart large object.</p> <p>For more information, see smart large objects, in the chapter on where data is stored in the <i>IBM Informix Administrator's Guide</i>. For information about obtaining the size of smart large objects, see the <i>IBM Informix DataBlade API Programmer's Guide</i> or the <i>IBM Informix ESQL/C Programmer's Manual</i>.</p>

This example creates a 20-megabyte mirrored sbspace, **eg_sbsp**, with the following specifications:

- An offset of 500 kilobytes for the primary and mirror chunks
- An offset of 200 kilobytes for the metadata area
- An average expected smart-large-object size of 32 kilobytes
- Log changes to the smart large objects in the user-data area of the sbspace

UNIX Only:

```
% onspaces -c -S eg_sbsp -p /dev/raw_dev1 -o 500 -s 20000
-m /dev/raw_dev2 500 -Mo 200 -Df "AVG_LO_SIZE=32,LOGGING=ON"
```

Changing the -Df Settings

As the database server administrator, you can override or change the **-Df** default settings in one of the following ways:

- To change the default settings for an sbspace, use the **onspaces -ch** option. For more information, refer to “onspaces -ch: Change sbspace default specifications” on page 19-17.
- To override the following **-Df** default settings for a specific table, use the SQL statements CREATE TABLE or ALTER TABLE:
 - LOGGING
 - ACESSTIME
 - EXTENT_SIZE
 - NEXT_SIZE

For more information on the ALTER TABLE and CREATE TABLE statements, see the *IBM Informix Guide to SQL: Syntax*.

The programmer can override these **-Df** default settings with DataBlade API and Informix ESQL/C functions. For information about altering storage characteristics of smart large objects, see the *IBM Informix DataBlade API Programmer's Guide* and the *IBM Informix ESQL/C Programmer's Manual*.

Using the onspaces -g option

The **onspaces -g** option is not used for sbspaces. The database server uses a different method to determine the number of pages to transfer in an I/O operation for sbspaces than for blobspaces. The database server can automatically determine the block size to transfer in an I/O operation for smart large objects. For more information, see sbpace extent sizes in the chapter on I/O activity in your *IBM Informix Performance Guide*.

This command has an equivalent SQL administration API function.

onspaces -c -x: Create an extspace

Use the **onspaces -c -x** option to create an extspace.

Syntax:

►► onspaces -c — -x—extspace— -l—location— -o—offset— -s—size— ◀◀

Element	Purpose	Key Considerations
-c	Creates a dbspace, blobspace, sbpace, or extspace You can create up to 2047 storage spaces of any type.	After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. For more information, see creating a dbspace, blobspace, or extspace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-l location	Specifies the location of the extspace The access method determines the format of this string.	Restrictions: String. Value must not be longer than 255 bytes. For more information, see creating an extspace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-o offset	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace, dbspace, or sbpace	Restrictions: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 terabytes, depending on the platform. For more information, see allocating raw disk space, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-s size	Indicates, in kilobytes, the size of the initial chunk of the new blobspace or dbspace	Restrictions: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.

Element	Purpose	Key Considerations
-x <i>extspace</i>	Names the extspace to be created	<p>Restrictions: Extspace names can be up to 128 bytes. They must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or \$ characters.</p> <p>For more information, see <i>extspaces</i>, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>.</p>

onspaces -ch: Change sbspace default specifications

Use the **onspaces -ch** option to change the default specifications of a sbspace.

Syntax:

►► **onspaces -ch—*sbspace*— -Df—*default list*** ◀◀

Element	Purpose	Key Considerations
-ch	Indicates that one or more sbspace default specifications are to be changed	None.
<i>sbspace</i>	Names the sbspace for which to change the default specifications	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For background information, see changing default specifications of an sbspace with onspaces in the <i>IBM Informix Performance Guide</i> .
-Df <i>default list</i>	Lists new default specifications for smart large objects stored in the sbspace	<p>Tags are separated by commas. If a tag is not present, system defaults take precedence. The list must be enclosed in double quotation marks (") on the command line.</p> <p>For a list of tags and their parameters, see Table 19-1 on page 19-13.</p>

You can change any of the **-Df** tags with the **onspaces -ch** option. The database server applies the change to each smart large object that was created prior to changing the default specification.

For example, to turn off logging for the sbspace that you created in “Creating an Sbspace with the -Df option” on page 19-12, use the following command:

```
onspaces -ch eg_sbsp -Df "LOGGING=OFF"
```

Note: After you turn on logging for an sbspace, take a level-0 backup of the sbspace to create a point from which to recover.

Related reference:

“set sbospace accesstime argument: Control access time tracking (SQL administration API)” on page 21-109

“set sbospace avg_lo_size argument: Set the average size of smart large objects (SQL administration API)” on page 21-110

“set sbospace logging argument: Change the logging of an sbospace (SQL administration API)” on page 21-111

onspaces -cl: Clean up stray smart large objects in sbospaces

Use the **onspaces -cl** option to clean up stray smart large objects in sbospaces.

Syntax:

►► onspaces -cl *sbspace* ◀◀

Element	Purpose	Key Considerations
-cl	Cleans up stray smart large objects in an sbospace	To find any stray smart large objects, use the oncheck -pS command when no users are connected to the database server. The smart large objects with a reference count of 0 are stray objects.
<i>sbspace</i>	Names the sbospace to be cleaned up	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> .

During normal operation, no unreferenced (stray) smart large objects should exist. When you delete a smart large object, the space is released. If the database server fails or runs out of system memory while you are deleting a smart large object, the smart large object might remain as a stray object.

The following is an example of the **onspaces -cl** command:

```
onspaces -cl myspace
```

The best way to find the reference count for a smart large object is to call the **mi_lo_stat** or **ifx_lo_stat** functions from a C program. Although the **mi_lo_increfcount** and **mi_lo_decrefcount** functions return the reference count, they increment or decrement the reference count. For more information on these functions, see the *IBM Informix DataBlade API Function Reference*.

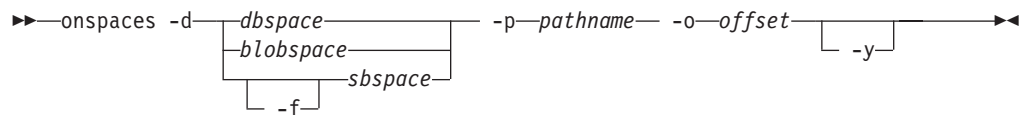
This command has an equivalent SQL administration API function.

Related reference:

“clean sbospace argument: Release unreferenced smart large objects (SQL administration API)” on page 21-26

onspaces -d: Drop a chunk in a dbspace, blobspace, or sbospace

Use the **onspaces -d** option to drop a chunk in a dbspace, blobspace, or sbospace.

Syntax:

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-d	Drops a chunk	<p>You can drop a chunk from a dbspace, temporary dbspace, or sbspace when the database server is online or quiescent. For more information, see the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>.</p> <p>You can drop a chunk from a blobspace only when the database server is in quiescent mode.</p>
-f	Drops an sbspace chunk that contains user data but no metadata. If the chunk contains metadata for the sbspace, you must drop the entire sbspace.	<p>Use the -f option with sbspaces only. If you omit the -f option, you cannot drop an sbspace that contains data.</p> <p>For more information, see dropping a chunk from an sbspace with onspaces, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>.</p>
-o offset	Indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the dbspace, blobspace, or sbspace that you are dropping	<p>Restrictions: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size.</p> <p>The maximum offset is 4 terabytes.</p> <p>For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>.</p>
-p pathname	Indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you are dropping	<p>The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server.</p> <p>For path name syntax, see your operating-system documentation.</p>
-y	Causes the database server to automatically respond yes to all prompts	None.
blobspace	Names the blobspace from which the chunk is dropped	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see dropping a chunk from a blobspace, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
dbspace	Names the dbspace from which the chunk is dropped	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see dropping a chunk from a dbspace with onspaces , in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .

Element	Purpose	Key considerations
<i>dbspace</i>	Names the dbspace to be dropped	Before you drop a dbspace, drop all databases and tables that you previously created in the dbspace.
<i>extspace</i>	Names the extspace to be dropped	You cannot drop an extspace if it is associated with an existing table or index.
<i>sbspace</i>	Names the sbspace to be dropped	Before you drop an sbspace, drop all tables that include a BLOB or CLOB column that references the sbspace.

Important: Do not specify a path name when you drop these storage spaces.

Related reference:

“drop blobspace argument: Drop a blobspace (SQL administration API)” on page 21-43

“drop dbspace argument: Drop a dbspace (SQL administration API)” on page 21-46

“drop sbspace argument: Drop an sbspace (SQL administration API)” on page 21-49

“drop tempdbspace argument: Drop a temporary dbspace (SQL administration API)” on page 21-50

onspaces -f: Specify DATASKIP parameter

Use the **onspaces -f** option to specify the value of the DATASKIP configuration parameter on a dbspace level or across all dbspaces.

Syntax:

```

>>> onspaces -f [OFF|ON] [dbspace-list] [-y]

```

This command has an equivalent SQL administration API function.

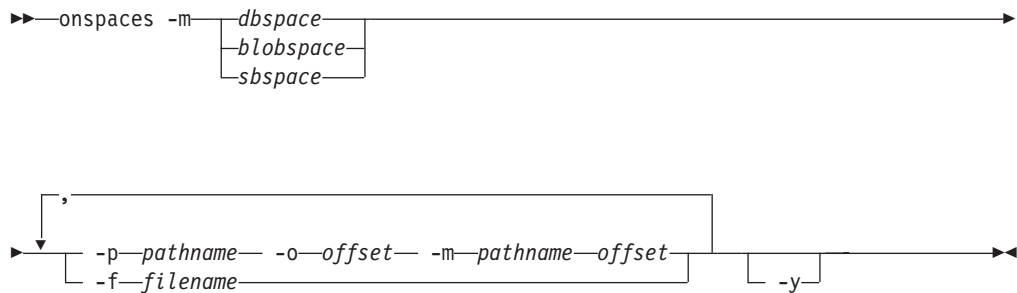
Element	Purpose	Key considerations
-f	Indicates to the database server that you want to change the DATASKIP default for specified dbspaces or all dbspaces	All changes in the DATASKIP status are recorded in the message log.
-y	Causes the database server to automatically respond yes to all prompts	None.
<i>dbspace-list</i>	Specifies the name of one or more dbspaces for which DATASKIP will be turned ON or OFF	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see “DATASKIP Configuration Parameter” on page 1-43 and the <i>IBM Informix Performance Guide</i> .
OFF	Turns off DATASKIP	If you use OFF without <i>dbspace-list</i> , DATASKIP is turned off for all fragments. If you use OFF with <i>dbspace-list</i> , only the specified fragments are set with DATASKIP off.
ON	Turns on DATASKIP	If you use ON without <i>dbspace-list</i> , DATASKIP is turned on for all fragments. If you use ON with <i>dbspace-list</i> , only the specified fragments are set with DATASKIP on.

Related reference:

“set dataskip argument: Start or stop skipping a dbspace (SQL administration API)” on page 21-106

onspaces -m: Start mirroring

Use the **onspaces -m** option to start mirroring for a dbspace, blobspace, or sbspace.

Syntax:

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-f filename	Indicates that chunk-location information is in a file named <i>filename</i>	<p>The file must be a buffered file that already exists. The path name must conform to the operating-system-specific rules for path names.</p> <p>For more information, see “Using a File to Specify Chunk-Location Information with the -f Option” on page 19-23.</p>
-m	Adds mirroring for an existing dbspace, blobspace, or sbspace	<p>User-data chunks in a mirrored sbspace need not be mirrored.</p> <p>The mirrored chunks should be on a different disk. You must mirror all the chunks at the same time.</p>
-m pathname offset	<p>The second time that <i>pathname</i> occurs in the syntax diagram, it indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that performs the mirroring.</p> <p>The second time <i>offset</i> appears in the syntax diagram, it indicates the offset to reach the mirrored chunk of the newly mirrored dbspace, blobspace, or sbspace. Also see the entries for <i>pathname</i> and <i>offset</i> in this table.</p>	None.

Element	Purpose	Key considerations
-o offset	The first time that <i>offset</i> occurs in the syntax diagram, it indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the newly mirrored dbspace, blobspace, or sbospace.	<p>Restrictions: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size.</p> <p>The maximum offset is 4 terabytes.</p> <p>For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>.</p>
-p pathname	The first time <i>pathname</i> occurs in the syntax diagram, it indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbospace that you want to mirror.	<p>The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server.</p> <p>For path name syntax, see your operating-system documentation.</p>
-y	Causes the database server to automatically respond yes to all prompts	None.
blobspace	Names the blobspace that you want to mirror	Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see the chapter on using mirroring in the <i>IBM Informix Administrator's Guide</i> .
dbspace	Names the dbspace that you want to mirror	Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> . For background information, see the chapter on using mirroring in the <i>IBM Informix Administrator's Guide</i> .
sospace	Names the sbospace that you want to mirror	Syntax must conform to the Identifier segment; see <i>IBM Informix Guide to SQL: Syntax</i> . For background information, see the chapter on using mirroring in the <i>IBM Informix Administrator's Guide</i> .

Related reference:

“add mirror argument: Add a mirror chunk (SQL administration API)” on page 21-18

“start mirroring argument: Starts storage space mirroring (SQL administration API)” on page 21-117

Using a File to Specify Chunk-Location Information with the -f Option

You can create a file that contains the chunk-location information. Then, when you execute **onspaces**, use the **-f** option to indicate to the database server that this information is in a file whose name you specify in *filename*.

The contents of the file should conform to the following format, with options separated by spaces and each set of primary and mirror chunks on separate lines:

```
primary_chunk_path offset mirror_chunk_path offset
```

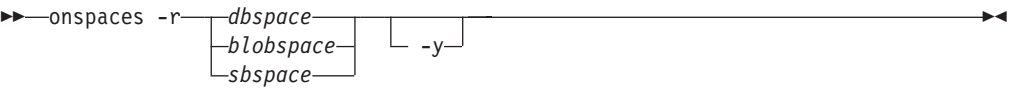
If the dbspace that you are mirroring contains multiple chunks, you must specify a mirror chunk for each of the primary chunks in the dbspace that you want to mirror. For an example that enables mirroring for a multichunk dbspace, see

starting mirroring for unmirrored dbspaces with **onspaces** in the chapter on using mirroring in the *IBM Informix Administrator's Guide*.

onspaces -r: Stop mirroring

Use the **onspaces -r** option to end mirroring for a dbspace, blobspace, or sbspace.

Syntax:



This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-r	Indicates to the database server that mirroring should be ended for an existing dbspace, blobspace, or sbspace	For background information, see the chapter on using mirroring in the <i>IBM Informix Administrator's Guide</i> .
-y	Causes the database server to respond yes to all prompts automatically	None.
<i>blobspace</i>	Names the blobspace for which you want to end mirroring.	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see the chapter on using mirroring in the <i>IBM Informix Administrator's Guide</i> .
<i>dbspace</i>	Names the dbspace for which you want to end mirroring.	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see the chapter on using mirroring in the <i>IBM Informix Administrator's Guide</i> .
<i>sbspace</i>	Names the sbspace for which you want to end mirroring	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For background information, see the chapter on using mirroring in the <i>IBM Informix Administrator's Guide</i> .

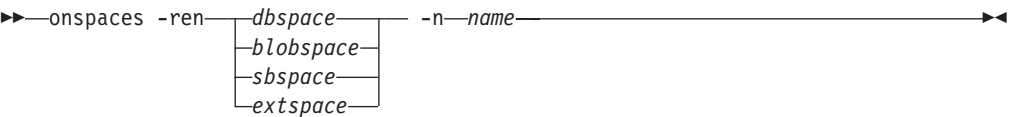
Related reference:

“stop mirroring argument: Stops storage space mirroring (SQL administration API)” on page 21-118

onspaces -ren: Rename a dbspace, blobspace, sbspace, or extspace

Use the **onspaces -ren** option to rename a dbspace, blobspace, sbspace, or extspace.

Syntax:



This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-ren	Causes the database server to rename the specified blob space, db space, ext space, or sb space	Restrictions: You can rename a blob space, db space, ext space, or sb space when the database server is in quiescent mode. For more information, see the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
-n name	Specifies the new name for the blob space, db space, ext space, or sb space	<p>Restrictions: The blob space, db space, external space, or sb space name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character.</p> <p>For more information, see the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>. The syntax must conform to the Identifier segment. For more information, see the <i>IBM Informix Guide to SQL: Syntax</i>.</p>
<i>blob space</i>	Names the blob space to be renamed	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see renaming spaces, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
<i>db space</i>	Names the db space to be renamed	<p>Restrictions: You cannot rename a critical db space, such as the root db space or a db space that contains physical logs.</p> <p>Additional Information: If you rename db spaces that are included in the DATASKIP list, update the DATASKIP configuration parameter with the new names using the onspaces -f command.</p> <p>Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i>. For more information, see renaming spaces, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>.</p>
<i>ext space</i>	Names the ext space to be renamed	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see renaming spaces, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .
<i>sb space</i>	Names the sb space to be renamed	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see renaming spaces, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i> .

Related reference:

"rename space argument: Rename a storage space (SQL administration API)" on page 21-98

Renaming a Db space, Blob space, Sb space, or Ext space when Enterprise Replication Is Active

You can rename a space (db space, blob space, sb space, or ext space) when Enterprise Replication is active. When you put the database server into quiescent mode to rename the space, Enterprise Replication will be disconnected. You can then rename the space. The servers will resynchronize after you put the database server into online mode. If you want to rename the same space on another server, you must put that server into quiescent mode and rename the space separately. No enforced relationship is propagated between renamed spaces on different ER servers; the same tables can be in different spaces.

If the Enterprise Replication server also participates in High-Availability Data Replication (HDR), you can rename the dbspace on the primary server and it will be automatically propagate to the secondary server. (The secondary server cannot participate in Enterprise Replication.)

Performing an Archive after Renaming a Space

After renaming any space (except extspaces or temporary spaces), perform a level-0 archive of the renamed space and the root dbspace. This will ensure that you can restore the spaces to a state including or following the rename dbspace operation. It is also necessary prior to performing any other type of archive.

onspaces -s: Change status of a mirrored chunk

Use the **onspaces -s** option to change the status of a mirrored chunk in a dbspace, a non-primary chunk within a noncritical dbspace, a blobspace, or an sbspace.

Syntax:

```

>> onspaces -s dbspace  
blobspace  
sbspace -p pathname -o offset -D  
-O -y

```

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-D	Indicates that you want to take the chunk down	None.
-o <i>offset</i>	Indicates, in kilobytes, the offset into the disk partition or unbuffered device to reach the chunk	<p>Restrictions: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The offset must be a multiple of the page size.</p> <p>The maximum offset is 4 terabytes.</p> <p>For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the <i>IBM Informix Administrator's Guide</i>.</p>
-O	Indicates that you want to restore the chunk and bring it online	None.
-p <i>pathname</i>	Indicates the disk partition or unbuffered device of the chunk	<p>The chunk can be an unbuffered device or a buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server.</p> <p>For path name syntax, see your operating-system documentation.</p>
-s	Indicates that you want to change the status of a chunk	<p>Restrictions: You can only change the status of a chunk in a mirrored pair or a non-primary chunk within a noncritical dbspace.</p> <p>For more information, see changing the mirror status in the <i>IBM Informix Administrator's Guide</i>.</p>

Element	Purpose	Key considerations
-y	Causes the database server to respond yes to all prompts automatically	None.
blobSPACE	Names the blobSPACE whose status you want to change	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see changing the mirror status in the <i>IBM Informix Administrator's Guide</i> .
dbSPACE	Names the dbSPACE whose status you want to change	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For more information, see changing the mirror status in the <i>IBM Informix Administrator's Guide</i> .
sbspace	Names the sbspace whose status you want to change	Syntax must conform to the Identifier segment; see the <i>IBM Informix Guide to SQL: Syntax</i> . For background information, see changing the mirror status in the <i>IBM Informix Administrator's Guide</i> .

Related reference:

“alter chunk argument: Change chunk status to online or offline (SQL administration API)” on page 21-19

“set chunk argument: Change the status of a chunk (SQL administration API)” on page 21-105

Avoid overwriting a chunk

The chunks associated with each Informix instance are not known to other Informix instances. It is possible to inadvertently create a chunk on a file or device that is allocated as a chunk to another Informix instance, which results in data corruption.

If you attempt to initialize an instance, where the ROOTPATH configuration parameter specifies a file or device that is the root chunk of another instance, the command fails with the following message in the online.log:

```
DISK INITIALIZATION ABORTED: potential instance overwrite detected.
```

To disable this initialization check, set the FULL_DISK_INIT configuration parameter to 1 in your configuration file and try to initialize the instance again. However, this initialization check is restricted to the root chunk. Adding dbSPACES or chunks succeeds even when the file or device is allocated to another instance.

Related reference:

"onspaces -a: Add a chunk to a dbspace or blobspace" on page 19-2

"onspaces -a: Add a chunk to an sbpace" on page 19-3

"onspaces -c -b: Create a blobspace" on page 19-5

"onspaces -c -d: Create a dbspace" on page 19-7

"onspaces -c -S: Create an sbpace" on page 19-11

"create blobspace argument: Create a blobspace (SQL administration API)" on page 21-26

"create chunk argument: Create a chunk (SQL administration API)" on page 21-29

"create dbspace argument: Create a dbspace (SQL administration API)" on page 21-32

"create sbpace argument: Create an sbpace (SQL administration API)" on page 21-34

Chapter 20. The onstat utility

The **onstat** utility reads shared-memory structures and provides statistics about the database server at the time that the command executes.

The *system-monitoring interface* also provides information about the database server. For information on the system-monitoring interface, see Chapter 2, “The sysmaster database,” on page 2-1.

You can combine multiple **onstat** option flags in a single command. The contents of shared memory might change as the **onstat** output displays. The **onstat** utility does not place any locks on shared memory, so running the utility does not affect performance.

These topics show you how to use the following **onstat** options:

- “**onstat** - command: Print output header” on page 20-23
- “**onstat** – command: Print onstat options and functions” on page 20-25
- “Running **onstat** Commands on a Shared Memory Dump File” on page 20-25
- “**onstat -a** command: Print overall status of the database server” on page 20-26
- “**onstat -b** command: Print buffer information for buffers in use” on page 20-26
- “**onstat -B** command: Prints information about used buffers” on page 20-27
- “**onstat -c** command: Print ONCONFIG file contents” on page 20-29
- “**onstat -C** command: Print B-tree scanner information” on page 20-29
- “**onstat -d** command: Print chunk information” on page 20-34
- “**onstat -D** command: Print page-read and page-write information” on page 20-39
- “**onstat -F** command: Print counts” on page 20-40
- “**onstat -g** monitoring options” on page 20-42
- “**onstat -G** command: Print TP/XA transaction information” on page 20-187
- “**onstat -h** command: Print buffer header hash chain information” on page 20-188
- “**onstat -j** command: Provide onpload status information” on page 20-190
- “**onstat -k** command: Print active lock information” on page 20-192
- “**onstat -l** command: Print physical and logical log information” on page 20-193
- “**onstat -o** command: Output shared memory contents to a file” on page 20-198
- “**onstat -p** command: Print profile counts” on page 20-200
- “**onstat -P** command: Print partition information” on page 20-203
- “**onstat -r** command: Repeatedly print selected statistics” on page 20-204
- “**onstat -t** and **onstat -T** commands: Print tblspace information” on page 20-211
- “**onstat -u** command: Print user activity profile” on page 20-212
- “**onstat -x** command: Print database server transaction information” on page 20-215
- “**onstat -X** command: Print thread information” on page 20-217
- “**onstat -z** command: Clear statistics” on page 20-219

onstat Portal: onstat Utility Commands Sorted by Functional Category

The information in this section lists **onstat** commands sorted by functional category.

Each category represents a different Informix feature for which **onstat** commands are useful for providing troubleshooting and performance enhancement information. Commands that appear in **bold** typeface are especially useful for providing troubleshooting information. Certain **onstat** commands are specific to one category, while others provide more general information and are listed in more than one category.

Category List

To use this section, you first determine the appropriate category from the following list, then follow the link to the **onstat** options for that category.

- “onstat Utility Archive Information Options”
- “onstat Utility Cache Information Options” on page 20-3
- “onstat Utility Debugging Options” on page 20-4
- “onstat Utility Enterprise Replication Options” on page 20-4
- “onstat Utility High-Availability Replication Options” on page 20-6
- “onstat Utility I/O Options” on page 20-7
- “onstat Utility Informix Warehouse Accelerator options” on page 20-8
- “onstat Utility Locks and Latches Options” on page 20-8
- “onstat Utility Logs Options” on page 20-8
- “onstat Utility Memory Options” on page 20-9
- “onstat Utility Network Options” on page 20-10
- “onstat Utility Performance Checks (First Tier)” on page 20-11
- “onstat Utility Performance Checks (Second Tier)” on page 20-12
- “onstat Utility Table Options” on page 20-13
- “onstat Utility Thread Options” on page 20-13
- “onstat Utility User/Session Options” on page 20-14
- “onstat Utility Virtual Processor Options” on page 20-15
- “onstat Utility Waiting Options” on page 20-15
- “onstat Utility compression options” on page 20-16
- “Other Useful onstat Utility Options” on page 20-16

onstat Utility Archive Information Options

Use the following **onstat** options to display information about archives and restores.

Table 20-1. onstat Utility Archive Information Options

Commands	Reference
onstat -D	Prints chunk I/O activity. Prints dbspace read/write activity for monitoring restore progress. “ onstat -D command: Print page-read and page-write information” on page 20-39

Table 20-1. *onstat* Utility Archive Information Options (continued)

Commands	Reference
onstat -g arc	Prints the last committed and any ongoing backups for each dbspace. “ onstat -g arc command: Print archive status” on page 20-52

onstat Utility Cache Information Options

Use the following **onstat** options to display information about caches and cached data, including buffer pools.

Table 20-2. *onstat* Utility Cache Information Options

Commands	Reference
onstat -b	Prints buffer pages in use. “ onstat -b command: Print buffer information for buffers in use” on page 20-26
onstat -B	Prints information about used buffers. “ onstat -B command: Prints information about used buffers” on page 20-27
onstat -F	Prints state of buffer queue cleaners and I/O. “ onstat -F command: Print counts” on page 20-40
onstat -g dic	Prints data dictionary cache, containing system catalog data for tables. Prints one line of information for each table cached in the shared-memory dictionary. For more information, see your <i>IBM Informix Performance Guide</i> . For sample output, see “ onstat -g dic command: Print table information” on page 20-76.
onstat -g dsc	Prints table distribution statistics for the optimizer. “ onstat -g dsc command: Print distribution cache information” on page 20-83.
onstat -g prc	Prints the stored procedure (SPL) routine cache. Prints information about SPL routine cache. “ onstat -g prc command: Print sessions using UDR or SPL routine” on page 20-134.
onstat -g ssc	Prints the number of times that the database server reads the SQL statement in the cache. Displays the same output as onstat -g cac . For more information, see improving query performance in the <i>IBM Informix Performance Guide</i> . “ onstat -g ssc command: Print SQL statement occurrences” on page 20-174.
onstat -g vpcache	Prints CPU virtual processor memory cache. “ onstat -g vpcache command: Print CPU VP memory block cache statistics” on page 20-182

Table 20-2. *onstat* Utility Cache Information Options (continued)

Commands	Reference
onstat -h	Prints buffer hash chain information. “ onstat -h command: Print buffer header hash chain information” on page 20-188
onstat -O	Prints optical subsystem memory cache and staging-area (disk cache) blobspace for TEXT or BYTE data. “ onstat -O command: Print optical subsystem information” on page 20-198
onstat -p	Prints global (server) information regarding the effectiveness of buffer pool caching. “ onstat -p command: Print profile counts” on page 20-200
onstat -X	Prints threads waiting for buffers. “ onstat -X command: Print thread information” on page 20-217

onstat Utility Debugging Options

Use the following **onstat** options to display information that is useful for debugging problems with the server.

Table 20-3. *onstat* Utility Debugging Options

Commands	Reference
onstat -g dmp	Prints raw memory at a given address for a number of given bytes. “ onstat -g dmp command: Print raw memory” on page 20-80
onstat -g src	Searches for patterns in shared memory. Note that memory is byte-swapped on Intel platforms. “ onstat -g src command: Patterns in shared memory” on page 20-173
onstat -o	Prints shared memory contents to a file. “ onstat -o command: Output shared memory contents to a file” on page 20-198.

onstat Utility Enterprise Replication Options

Use the following **onstat** options to track Enterprise Replication statistics and to provide troubleshooting information. For additional information about Enterprise Replications see the **cdr view** and **cdr view profile** commands described in the *IBM Informix Enterprise Replication Guide*.

Table 20-4. *onstat Utility Enterprise Replication Options*

Commands	Reference
onstat -g cat	Prints information from the Enterprise Replication global catalog. The global catalog contains a summary of information about the defined servers, replicates, and replicate sets on each of the servers within the enterprise. “ onstat -g cat command: Print ER global catalog information” on page 20-57.
onstat -g cdr	Prints the settings of Enterprise Replication configuration parameters and environment variables. “ onstat -g cdr config command: Print ER settings” on page 20-59.
onstat -g cdr config	Prints Enterprise Replication configuration parameters and environment variables. “ onstat -g cdr config command: Print ER settings” on page 20-59
onstat -g ddr	Prints status of Enterprise Replication components that read and process log records. “ onstat -g ddr command: Print ER database log reader status” on page 20-74
onstat -g dss	Prints activity of individual data sync (transaction processing) threads. “ onstat -g dss command: Print ER environment data” on page 20-84
onstat -g dtc	Prints delete table cleaner activity. Deleted or updated rows placed in the delete table are purged at intervals. “ onstat -g dtc command: Print delete table cleaner statistics” on page 20-85
onstat -g grp	Prints Enterprise Replication grouper statistics. The grouper evaluates the log records, rebuilds the individual log records into the original transaction, packages the transaction, and queues the transaction for transmission. “ onstat -g grp command: Print ER grouper statistics” on page 20-91
onstat -g nif	Prints network interface statistics. Shows the state of the network interface, servers, and data transfer among servers. “ onstat -g nif command: Print statistics about the network interface” on page 20-120
onstat -g que	Prints statistics for the high-level queue interface (which is common to all of the queues of the Enterprise Replication Queue Manager). “ onstat -g que command: Prints ER queue statistics” on page 20-140
onstat -g rcv	Prints receive manager statistics. “ onstat -g rcv command: Print ER receive manager statistics” on page 20-144

Table 20-4. *onstat Utility Enterprise Replication Options (continued)*

Commands	Reference
onstat -g rep	Prints events that are in the queue for the schedule manager. “ onstat -g rep command: Print ER schedule manager events” on page 20-148
onstat -g rqm	Prints statistics and contents of the low-level queues (send queue, receive queue, ack send queue, sync send queue, and control send queue) managed by the Reliable Queue Manager (RQM). “ onstat -g rqm command: Print low-level queue statistics” on page 20-148
onstat -g sync	Prints synchronization status. “ onstat -g sync command: Print ER synchronization status” on page 20-179

onstat Utility High-Availability Replication Options

Use the following **onstat** options to monitor high-availability cluster environments and the Connection Manager.

Table 20-5. *onstat Utility High-Availability Replication Options*

Commands	Reference
onstat -g cluster	Prints high-availability cluster information. “ onstat -g cluster command: Print high-availability cluster information” on page 20-65
onstat -g cmsm	Prints Connection Manager information. “ onstat -g cmsm command: Print Connection Manager information” on page 20-69
onstat -g dri	Prints data-replication information. See <i>Monitoring High-Availability Data-Replication status</i> in the <i>IBM Informix Administrator's Guide</i> . “ onstat -g dri command: Print high-availability data replication information” on page 20-81.
onstat -g ipl	Prints index page logging status. “ onstat -g ipl command: Print index page logging status information” on page 20-107
onstat -g laq	Prints information about queues on the secondary server. “ onstat -g laq command: Print secondary server queues” on page 20-109
onstat -g proxy	Prints proxy distributors for high-availability. “ onstat -g proxy command: Print proxy distributor information” on page 20-135
onstat -g rss	Prints remote stand-alone server (RSS) information. “ onstat -g rss command: Print RS secondary server information” on page 20-151

Table 20-5. *onstat Utility High-Availability Replication Options (continued)*

Commands	Reference
onstat -g sds	Prints shared disk secondary (SDS) server information. “ onstat -g sds command: Print SD secondary server information” on page 20-157
onstat -g smx	Prints Server Multiplexer Group (SMX) connections in high-availability environments. Prints data transfer statistics and encryption status. “ onstat -g smx command: Print multiplexer group information” on page 20-169

onstat Utility I/O Options

Use the following **onstat** options to track input and output (read and write) activity.

Table 20-6. *onstat Utility I/O Options*

Commands	Reference
onstat -D	Prints chunk I/O activity. “ onstat -D command: Print page-read and page-write information” on page 20-39
onstat -g cpu	Prints runtime statistics for each thread. “ onstat -g cpu : Print runtime statistics” on page 20-71
onstat -g ioa	Prints combined information from onstat -g ioq (queues), onstat -g iov (virtual processors), and onstat -g iob (big buffer). “ onstat -g ioa command: Print combined onstat -g information” on page 20-102
onstat -g iob	Prints the big buffer usage summary. “ onstat -g iob command: Print big buffer use summary” on page 20-103
onstat -g iof	Prints I/O statistics by file or chunk. This option is similar to the onstat -D option, but also displays information about non-chunk, temporary, and sort-work files. “ onstat -g iof command: Print asynchronous I/O statistics” on page 20-104
onstat -g iog	Prints AIO global information. “ onstat -g iog command: Print AIO global information” on page 20-105
onstat -g ioq	Prints queue read/write statistics and queue length. “ onstat -g ioq command: Print I/O queue information” on page 20-105. Also see the <i>IBM Informix Performance Guide</i> .
onstat -g iov	Prints asynchronous I/O statistics by virtual processor. “ onstat -g iov command: Print AIO VP statistics” on page 20-107

Table 20-6. *onstat Utility I/O Options (continued)*

Commands	Reference
onstat -p	Prints global disk activity including sequential scans. “ onstat -p command: Print profile counts” on page 20-200

onstat Utility Informix Warehouse Accelerator options

Use the following **onstat** options to display information exchanged between the database server and the Informix Warehouse Accelerator.

Table 20-7. *onstat Utility Informix Warehouse Accelerator options*

Commands	Reference
onstat -g aqt	Prints information about the data marts and the associated accelerated query tables (AQTs). “ onstat -g aqt command: Print data mart and accelerated query table information” on page 20-49.

onstat Utility Locks and Latches Options

Use the following **onstat** options to display information about locks.

Table 20-8. *onstat Utility Locks and Latches Options*

Commands	Reference
onstat -k	Prints information about active locks. “ onstat -k command: Print active lock information” on page 20-192
onstat -L	Prints the number of locks on a lock free list. “ onstat -L command: Print the number of free locks” on page 20-196
onstat -p	Prints global statistics on lock requests, lock waits, and latch waits. “ onstat -p command: Print profile counts” on page 20-200
onstat -s	Prints latch (mutex) information. “ onstat -s command: Print latch information” on page 20-209

onstat Utility Logs Options

Use the following **onstat** options to monitor logical and physical logs.

Table 20-9. *onstat Utility Logs Options*

Commands	Reference
onstat -g ipl	Prints index page logging information in high-availability environments. “ onstat -g ipl command: Print index page logging status information” on page 20-107

Table 20-9. *onstat* Utility Logs Options (continued)

Commands	Reference
<code>onstat -l</code>	Prints status of physical and logical logs, and log buffering. “ onstat -l command: Print physical and logical log information” on page 20-193

onstat Utility Memory Options

Use the following **onstat** options to monitor the various aspects of server memory allocation and use.

Table 20-10. *onstat* Utility Memory Options

Commands	Reference
<code>onstat -g afr</code>	Prints allocated memory fragments for a specified session or shared-memory pool. To obtain the pool name, see the onstat -g mem option. “ onstat -g afr command: Print allocated memory fragments” on page 20-48
<code>onstat -g ffr</code> (<i>pool name session ID</i>)	Prints free fragments for a session or shared memory pool. “ onstat -g ffr command: Print free fragments” on page 20-88
<code>onstat -g lmm</code>	Prints information about automatic low memory management settings and recent activity: “ onstat -g lmm command: Print low memory management information” on page 20-111
<code>onstat -g mem</code>	Prints session or pool virtual shared memory statistics. “ onstat -g mem command: Print pool memory statistics” on page 20-114
<code>onstat -g mgm</code>	Prints Memory Grant Manager (parallel and sort operations) resource information. “ onstat -g mgm command: Print MGM resource information” on page 20-116. Also see the <i>IBM Informix Performance Guide</i> .
<code>onstat -g nbm</code>	Prints block map for non-resident segments. “ onstat -g nbm command: Print a block bit map” on page 20-119
<code>onstat -g rbm</code>	Prints block map for resident segment. “ onstat -g rbm command: Print a block map of shared memory” on page 20-143
<code>onstat -g seg</code>	Prints memory segment statistics. “ onstat -g seg command: Print shared memory segment statistics” on page 20-161. Also see the <i>IBM Informix Administrator's Guide</i> .
<code>onstat -g ses</code>	Prints session information, including memory breakdown. For detailed information, use: onstat -g ses session_id “ onstat -g ses command: Print session-related information” on page 20-162 Also see the <i>IBM Informix Performance Guide</i>

Table 20-10. *onstat Utility Memory Options (continued)*

Commands	Reference
onstat -g stm	Prints SQL statement memory use. “ onstat -g stm command: Print SQL statement memory usage” on page 20-176
onstat -g stq	Prints stream queue buffers. “ onstat -g stq command: Print queue information” on page 20-177
onstat -g ufr	Prints memory pool fragments for a session or shared memory pool in use. “ onstat -g ufr command: Print memory pool fragments” on page 20-181
onstat -R	Prints buffer pool queues and their status. “ onstat -R command: Print LRU, FLRU, and MLRU queue information” on page 20-207

onstat Utility Network Options

Use the following **onstat** options to monitor shared memory and network connection services.

Table 20-11. *onstat Utility Network Options*

Commands	Reference
onstat -g imc	Prints information about Informix MaxConnect instances that are connected to the database server. If Informix MaxConnect is not connected to the database server, this command displays <i>No MaxConnect servers are connected</i> .
onstat -g nsc	Prints shared-memory status by <i>client id</i> . If <i>client id</i> is omitted, all client status areas are displayed. This command prints the same status data as the nss command. “ onstat -g nsc command: Print current shared memory connection information” on page 20-121
onstat -g nsd	Prints network shared-memory data for poll threads. “ onstat -g nsd command: Print poll threads shared-memory data” on page 20-124
onstat -g nss	Prints network shared-memory status by <i>session id</i> . If <i>session id</i> is omitted, all session status areas are displayed. This command prints the same status data as the onstat -g nsc command. “ onstat -g nss command: Print shared memory network connections status” on page 20-124
onstat -g nta	Prints combined network statistics from onstat -g ntd , onstat -g ntm , onstat -g ntt , and onstat -g ntu . If Informix MaxConnect is installed, this command prints statistics that you can use to tune Informix MaxConnect performance.
onstat -g ntd	Prints network statistics by service. “ onstat -g ntd command: Print network statistics” on page 20-125

Table 20-11. *onstat* Utility Network Options (continued)

Commands	Reference
onstat -g ntm	Prints network mail statistics. “ onstat -g ntm command: Print network mail statistics” on page 20-126
onstat -g ntt	Prints network user times. “ onstat -g ntt command: Print network user times” on page 20-126
onstat -g ntu	Prints network user statistics. “ onstat -g ntu command: Print network user statistics” on page 20-127

onstat Utility Performance Checks (First Tier)

Use the following **onstat** options to monitor performance and to check for performance impediments. Use the second-tier **onstat** options (and other **onstat** commands) to further narrow the problem.

Table 20-12. *onstat* Utility Performance Checks (First Tier)

Commands	Reference
onstat -c	Prints server configuration. “ onstat -c command: Print ONCONFIG file contents” on page 20-29
onstat -D	Prints chunk I/O. “ onstat -D command: Print page-read and page-write information” on page 20-39
onstat -g ath	Prints status and statistics for all threads. The sqlexec thread is a client session thread. The rstcb value corresponds to the user field of the onstat -u command. “ onstat -g ath command: Print information about all threads” on page 20-54. For information about using onstat -g ath to print Enterprise Replication threads, see the <i>IBM Informix Enterprise Replication Guide</i> .
onstat -g ckp	Prints checkpoint history and display configuration recommendations. “ onstat -g ckp command: Print checkpoint history and configuration recommendations” on page 20-61
onstat -g cpu	Prints runtime statistics for each thread. “ onstat -g cpu : Print runtime statistics” on page 20-71
onstat -g ioq	Prints pending I/O operations for the <i>queue name</i> . “ onstat -g ioq command: Print I/O queue information” on page 20-105
onstat -p	Prints global server performance profile. “ onstat -p command: Print profile counts” on page 20-200

Table 20-12. *onstat* Utility Performance Checks (First Tier) (continued)

Commands	Reference
onstat -u	Prints status and statistics for user threads. If a thread is waiting for a resource, this command identifies the type (flags field) and address (wait field) of the resource. “ onstat -u command: Print user activity profile” on page 20-212

onstat Utility Performance Checks (Second Tier)

Use the following **onstat** options to identify performance impediments.

Table 20-13. *onstat* Utility Performance Checks (Second Tier)

Commands	Reference
onstat -b	Prints active buffers. “ onstat -b command: Print buffer information for buffers in use” on page 20-26
onstat -g act	Prints active threads. “ onstat -g act command: Print active threads” on page 20-48
onstat -g glo	Prints virtual processors and their operating system processes (oninit processes). Prints virtual processor CPU use. On Windows, the virtual processors are operating system threads, and the values in the pid field are thread IDs. “ onstat -g glo command: Print global multithreading information” on page 20-89
onstat -g mgm	Prints Memory Grant Manager resource information. “ onstat -g mgm command: Print MGM resource information” on page 20-116
onstat -g rah	Prints read-ahead request information “ onstat -g rah command: Print read-ahead request statistics” on page 20-142
onstat -g rea	Prints threads in the ready queue waiting for CPU resources. “ onstat -g rea command: Print ready threads” on page 20-147
onstat -g seg	Prints shared-memory-segment statistics. This option shows the number and size of shared-memory segments allocated to the database server. “ onstat -g seg command: Print shared memory segment statistics” on page 20-161.
onstat -g wai	Prints waiting threads; all threads waiting for mutex or condition, or yielding. “ onstat -g wai command: Print wait queue thread list” on page 20-184
onstat -k	Prints active locks. “ onstat -k command: Print active lock information” on page 20-192

onstat Utility Table Options

Use the following **onstat** options to display information about table status and table statistics.

Table 20-14. *onstat* Utility Table Options

Commands	Reference
onstat -g buf	Prints buffer pool profile information. “ onstat -g buf command: Print buffer pool profile information” on page 20-55
onstat -g lap	Prints information about the status of currently active light appends (writes bypassing the buffer pool). “ onstat -g lap command: Print light appends status information” on page 20-109
onstat -g opn	Prints open partitions (tables). “ onstat -g opn command: Print open partitions” on page 20-127
onstat -g ppf	Prints partition profile (activity data) for the specified partition number or prints profiles for all partitions. “ onstat -g ppf command: Print partition profiles” on page 20-132
onstat -g scn	Prints information about the progress of a scan, based on rows scanned on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data, and identifies whether a scan is a light or bufferpool scan. “ onstat -g scn command: Print scan information” on page 20-155
onstat -P	Prints table and B-tree pages in the buffer pool, listed by partition (table). “ onstat -P command: Print partition information” on page 20-203
onstat -t onstat -T	Prints basic tblspace (partition) information for active (t) or all (T) tblspaces. “ onstat -t and onstat -T commands: Print tblspace information” on page 20-211

onstat Utility Thread Options

Use the following **onstat** options to display the status and activity of threads.

Table 20-15. *onstat* Utility Thread Options

Commands	Reference
onstat -g act	Prints active threads. This output is included in onstat -g ath output. “ onstat -g act command: Print active threads” on page 20-48
onstat -g ath	Prints all threads. “ onstat -g ath command: Print information about all threads” on page 20-54. For information about using onstat -g ath to print Enterprise Replication threads, see the <i>IBM Informix Enterprise Replication Guide</i> .

Table 20-15. *onstat* Utility Thread Options (continued)

Commands	Reference
onstat -g cpu	Prints runtime statistics for each thread. “ onstat -g cpu : Print runtime statistics” on page 20-71
onstat -g rea	Prints ready threads (threads waiting for CPU resource). This output is included in the onstat -g ath output. “ onstat -g rea command: Print ready threads” on page 20-147.
onstat -g sle	Prints information about threads sleeping for a specified time. Does not include threads that are sleeping forever. “ onstat -g sle command: Print all sleeping threads” on page 20-167
onstat -g stk	Prints the stack of a specified thread or prints stacks for all threads. “ onstat -g stk command: Print thread stack” on page 20-176
onstat -g sts	Prints maximum and current stack use per thread. “ onstat -g sts command: Print stack usage for each thread” on page 20-177
onstat -g tpf	Prints thread activity statistics. “ onstat -g tpf command: Print thread profiles” on page 20-180
onstat -g wai	Prints waiting (idle, sleeping, and waiting) threads. Included in onstat -g ath output. “ onstat -g wai command: Print wait queue thread list” on page 20-184
onstat -g wst	Prints wait statistics for threads. “ onstat -g wst command: Print wait statistics for threads” on page 20-185

onstat Utility User/Session Options

Use the following **onstat** options to display information about the user environment and active sessions.

Table 20-16. *onstat* Utility User/Session Options

Commands	Reference
onstat -g env	Prints the values of environment variables the database server is currently using. “ onstat -g env command: Print environment variable values” on page 20-86
onstat -g his	Prints SQL tracing information. “ onstat -g his command: Print SQL trace information” on page 20-97
onstat -g pqs	Prints operators used in currently running SQL queries. “ onstat -g pqs command: Print operators for all SQL queries” on page 20-133

Table 20-16. *onstat Utility User/Session Options (continued)*

Commands	Reference
onstat -g ses	Prints summary information for all active sessions or detailed information for individual sessions. “ onstat -g ses command: Print session-related information” on page 20-162
onstat -g sql	Prints SQL information for all active sessions or detailed SQL information for individual sessions. “ onstat -g sql command: Print SQL-related session information” on page 20-172
onstat -G	Prints global transactions. “ onstat -G command: Print TP/XA transaction information” on page 20-187
onstat -u	Prints status of user threads and their global read/write statistics. “ onstat -u command: Print user activity profile” on page 20-212
onstat -x	Prints information about transactions. “ onstat -x command: Print database server transaction information” on page 20-215

onstat Utility Virtual Processor Options

Use the following **onstat** options to display information and statistics for virtual processors.

Table 20-17. *onstat Utility Virtual Processor Options*

Commands	Reference
onstat -g glo	Prints global multithreading information and global statistics for virtual processor classes and individual virtual processors. On Windows, the virtual processors are operating system threads, and the values in the pid field are thread IDs. “ onstat -g glo command: Print global multithreading information” on page 20-89
onstat -g sch	Prints the number of semaphore operations, spins, and busy waits for each virtual processor. On Windows, the virtual processors are operating system threads, and the values in the pid field are thread IDs. “ onstat -g sch command: Print VP information” on page 20-155

onstat Utility Waiting Options

Use the following **onstat** options to display information about wait conditions for threads.

Table 20-18. *onstat Utility Waiting Options*

Commands	Reference
onstat -g con	Prints IDs of threads waiting for conditions. onstat -g ath to print thread information. See “ onstat -g con command: Print condition and thread information” on page 20-71
onstat -g lmx	Prints all locked mutexes. “ onstat -g lmx command: Print all locked mutexes” on page 20-113
onstat -g qst	Prints queue wait statistics for mutex and condition queues. “ onstat -g qst command: Print wait options for mutex and condition queues” on page 20-141
onstat -g rwm	Prints read/write mutexes. “ onstat -g rwm command: Print read and write mutexes” on page 20-154
onstat -g spi	Prints spin locks with long spins and spin lock statistics. “ onstat -g spi command: Print spin locks with long spins” on page 20-170
onstat -g wai	Prints waiting threads; all threads waiting for mutex or condition, or yielding. “ onstat -g wai command: Print wait queue thread list” on page 20-184
onstat -g wmx	Prints all mutexes with waiters. “ onstat -g wmx command: Print all mutexes with waiters” on page 20-184

onstat Utility compression options

Table 20-19. *onstat Utility compression Options*

Commands	Reference
onstat -g dsk	Prints progress of currently running compression operations. “ onstat -g dsk command: Print the progress of the currently running compression operation” on page 20-78
onstat -g ppd	Prints partition compression dictionary information. “ onstat -g ppd command: Print partition compression dictionary information” on page 20-131

Other Useful onstat Utility Options

Table 20-20. *Other Useful onstat Utility Options*

Commands	Reference
onstat -	Prints onstat header; includes engine version, status (On-Line, Quiescent, and so on), elapsed time since initialization, and memory footprint. “ onstat - command: Print output header” on page 20-23

Table 20-20. Other Useful onstat Utility Options (continued)

Commands	Reference
onstat -	Prints onstat usage options. “ onstat - command: Print onstat options and functions” on page 20-25
onstat options infile	Print onstat output using a shared memory dump (infile) as input. “Running onstat Commands on a Shared Memory Dump File” on page 20-25
onstat -a	Prints collective onstat outputs. “ onstat -a command: Print overall status of the database server” on page 20-26
onstat -c	Prints the server configuration file. “ onstat -c command: Print ONCONFIG file contents” on page 20-29
onstat -C	Prints B-tree index scanner information (shows statistics about index cleaning). “ onstat -C command: Print B-tree scanner information” on page 20-29
onstat -d	Prints chunk information. “ onstat -d command: Print chunk information” on page 20-34
onstat -f	Prints dbspaces configured for dataskip. “ onstat -f command: Print dbspace information affected by dataskip” on page 20-40
onstat -g all	Prints diagnostic information. “ onstat -g all command: Print diagnostic information” on page 20-49
onstat -g dbc	Prints statistics about dbScheduler and dbWorker threads. “ onstat -g dbc command: Print dbScheduler and dbWorker thread statistics” on page 20-73
onstat -g dis	Prints a list of database servers, their status, directory location, configuration information, and host name. “ onstat -g dis command: Print database server information” on page 20-77
onstat -g dll	Prints a list of dynamic libraries that have been loaded. “ onstat -g dll command: Print dynamic link library file list” on page 20-79
onstat -g osi	Prints information about operating system resources and parameters. “ onstat -g osi : Print operating system information” on page 20-129

Table 20-20. Other Useful onstat Utility Options (continued)

Commands	Reference
onstat -g pos	Prints values from <code>\$INFORMIXDIR/etc/infos.servernum</code> file, which are used by clients such as onmode for shared memory connections to the server. onmode -R rebuilds the <code>\$INFORMIXDIR/etc/infos.servernum</code> file. “ onstat -g pos command: Print file values” on page 20-130
onstat -g smb	Prints detailed information about sbspaces. “ onstat -g smb command: Print sbspaces information” on page 20-167
onstat -g sym	Prints symbol table information for the oninit utility. “ onstat -g sym command: Print symbol table information for the oninit utility” on page 20-178
onstat -i	Changes onstat mode to interactive. “ onstat -i command: Initiate interactive mode” on page 20-189
onstat -j	Prints information about the status of an onpload job. “ onstat -j command: Provide onpload status information” on page 20-190
onstat -m	Prints message log contents. “ onstat -m command: Print recent system message log information” on page 20-197
onstat -O	Prints Optical subsystem cache information. “ onstat -O command: Print optical subsystem information” on page 20-198
onstat -r	Prints repetitive onstat execution. “ onstat -r command: Repeatedly print selected statistics” on page 20-204
onstat -z	Resets the accumulated statistics to zero. “ onstat -z command: Clear statistics” on page 20-219

Monitor the database server status

A useful feature of the output of the **onstat** command is the heading that indicates the database server status.

Whenever the database server is blocked, **onstat** displays the following line after the banner line:

Blocked: *reason*

The variable *reason* can be one or more of the following values.

Reason	Description
ADMINISTRATION	Database is in administration mode
ARCHIVE	Ongoing storage-space backup
ARCHIVE_EBR	Blocked for External Backup and Recovery.

Reason	Description
CHG_PLOG	Blocked while physical log is being changed.
CKPT	Checkpoint
CKPT INP	Interval checkpoint in progress
DBS_DROP	Dropping a dbspace
DDR	Discrete data replication
DYNAMIC_LOG	Log file is being added dynamically
DYNAMIC_LOG_FOR_ER	Log file is being added dynamically in ER setup
FREE_LOG	Log file is being freed
HA_CONV_STD	Blocked while High Availability server is being converted to standard server.
HA_FAILOVER	Blocked while High Availability server failover being processed.
HANG_SYSTEM	Database server failure
LAST_LOG_RESERVED4BACKUP	Waiting for last available log to be backed up
LBU	Logs full high-watermark
LOG_DROP	Log file is being dropped
LONGTX	Long transaction
MEDIA_FAILURE	Media failure
OVERRIDE_DOWN_SPACE	Waiting to override down dbspace setting because the ONDBSPACEDOWN onconfig parameter is set to WAIT

In this table, the value CHKP INP does not indicate that the database server is blocked, but that a nonblocking interval checkpoint is in progress while the buffer pool is being flushed. This CHKP INP value appears in the status line of **onstat** output until all pages in the shared-memory buffer pool have been written to disk. For information about setting interval checkpoints to flush the buffer pool, see the “CKPTINTVL configuration parameter” on page 1-41.

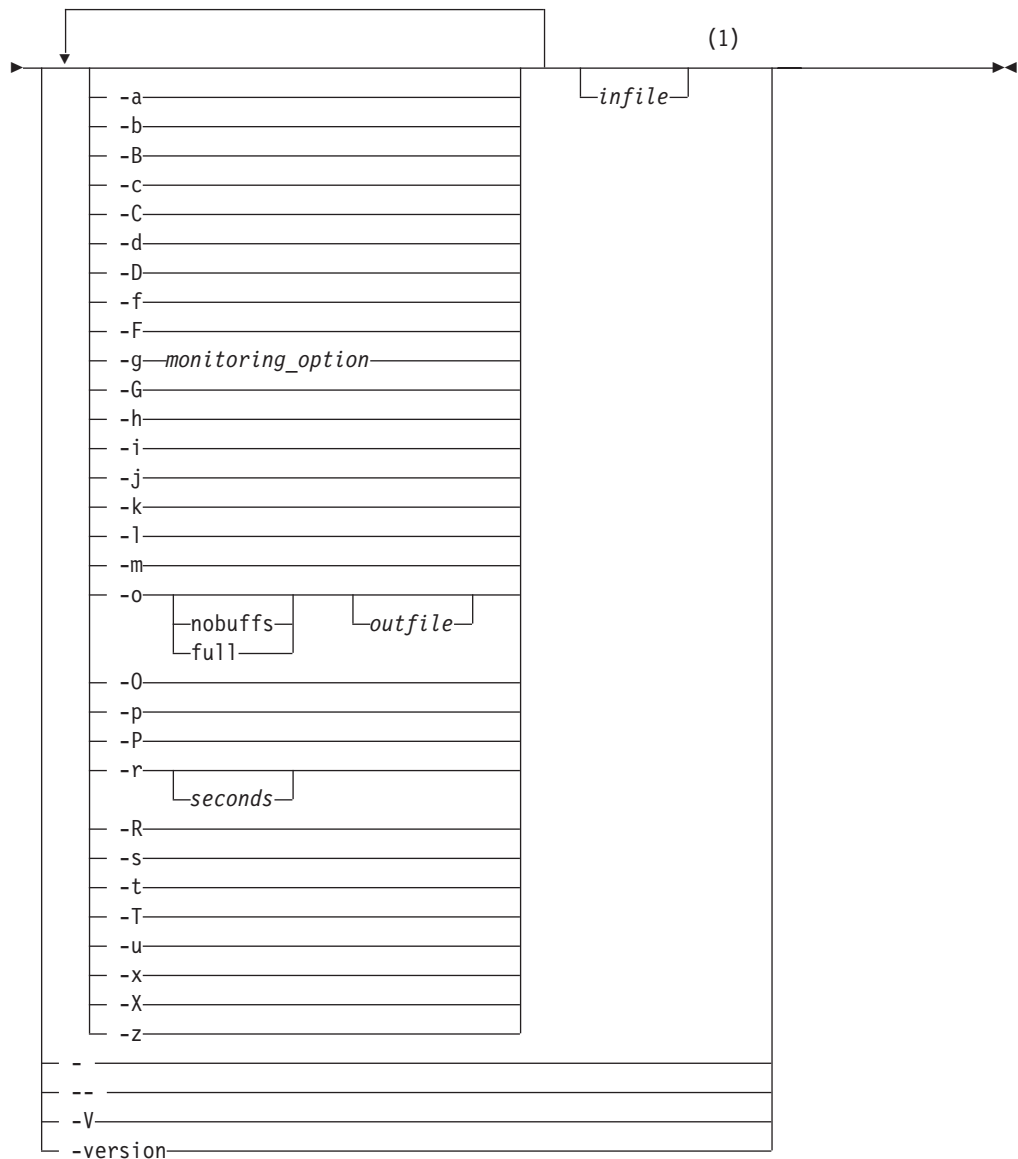
onstat Command Syntax

The complete syntax for the **onstat** command, including information about the interactive mode and how to have options to execute repeatedly.

```

▶▶ onstat [-pu]

```



Notes:

- 1 Only one occurrence of each item is allowed. More than one option can be specified on a single **onstat** command invocation.

Element	Purpose	Key Considerations
-	Displays the output header only.	See “ onstat - command: Print output header” on page 20-23.
--	Displays a listing of all onstat options and their functions	See “ onstat -- command: Print onstat options and functions” on page 20-25. This option cannot be combined with any other onstat option.
-a	Interpreted as onstat -cuskbtdlp . Displays output in that order.	See “ onstat -a command: Print overall status of the database server” on page 20-26.

Element	Purpose	Key Considerations
-b	Displays information about buffers currently in use, including number of resident pages in the buffer pool	See “ onstat -b command: Print buffer information for buffers in use” on page 20-26.
-B	Obtains information about all database server buffers, not just buffers currently in use.	See “ onstat -B command: Prints information about used buffers” on page 20-27.
-c	Displays the ONCONFIG file: <ul style="list-style-type: none"> • \$INFORMIXDIR/etc/\$ONCONFIG for UNIX • %INFORMIXDIR%\etc\%ONCONFIG% for Windows 	See “ onstat -c command: Print ONCONFIG file contents” on page 20-29.
-C	Prints B-tree scanner information	See “ onstat -C command: Print B-tree scanner information” on page 20-29.
-d	Displays information for chunks in each storage space	See “ onstat -d command: Print chunk information” on page 20-34.
-D	Displays page-read and page-write information for the first 50 chunks in each dbspace	See “ onstat -D command: Print page-read and page-write information” on page 20-39.
-f	Lists the dbspaces currently affected by the DATASKIP feature	See “ onstat -f command: Print dbspace information affected by dataskip” on page 20-40.
-F	Displays a count for each type of write that flushes pages to disk	See “ onstat -F command: Print counts” on page 20-40.
-g option	Prints monitoring option	See “ onstat -g monitoring options” on page 20-42.
-G	Prints global transaction IDs	See “ onstat -G command: Print TP/XA transaction information” on page 20-187.
-h	Provides information on the buffer header hash chains	See “ onstat -h command: Print buffer header hash chain information” on page 20-188.
-i	Puts the onstat utility into interactive mode	See “ onstat -i command: Initiate interactive mode” on page 20-189.
-j	Prints the interactive status of the active onpload process	See “ onstat -j command: Provide onpload status information” on page 20-190.
-k	Displays information about active locks	See “ onstat -k command: Print active lock information” on page 20-192.
-l	Displays information about physical and logical logs, including page addresses	See “ onstat -l command: Print physical and logical log information” on page 20-193.
-m	Displays the 20 most recent lines of the database server message log	Output from this option lists the full pathname of the message-log file and the 20 file entries. A date-and-time header separates the entries for each day. A time stamp prefates single entries within each day. The name of the message log is specified as MSGPATH in the ONCONFIG file. See “ onstat -m command: Print recent system message log information” on page 20-197.
-o	Saves a copy of the shared-memory segments to <i>outfile</i>	See “ onstat -o command: Output shared memory contents to a file” on page 20-198.
-O	Displays information about the Optical Subsystem memory cache and staging-area blob space	See “ onstat -O command: Print optical subsystem information” on page 20-198.

Element	Purpose	Key Considerations
-p	Displays profile counts.	See “ onstat -p command: Print profile counts” on page 20-200.
-P	Displays for all partitions the partition number and the break-up of the buffer-pool pages that belong to the partition	See “ onstat -P command: Print partition information” on page 20-203.
-pu	If you invoke onstat without any options, the command is interpreted as onstat -pu (-p option and -u option). Displays profile counts and prints a profile of user activity	See “ onstat -p command: Print profile counts” on page 20-200 and “ onstat -u command: Print user activity profile” on page 20-212.
-r seconds	Repeats the accompanying onstat options after a wait time specified in <i>seconds</i> between each execution	See “ onstat -r command: Repeatedly print selected statistics” on page 20-204.
-R	Displays detailed information about the LRU queues, FLRU queues, and MLRU queues	See “ onstat -R command: Print LRU, FLRU, and MLRU queue information” on page 20-207.
-s	Displays general latch information	See “ onstat -s command: Print latch information” on page 20-209.
-t	Displays tbspace information, including residency state, for active tbspaces	See “ onstat -t and onstat -T commands: Print tbspace information” on page 20-211.
-T	Displays tbspace information for all tbspaces	See “ onstat -t and onstat -T commands: Print tbspace information” on page 20-211.
-u	Prints a profile of user activity	See “ onstat -u command: Print user activity profile” on page 20-212.
-V	Displays the software version number and the serial number. This option cannot be combined with any other onstat option.	See “Obtaining Utility Version Information” on page 6-1.
-version	Displays the build version, host, OS, number and date, as well as the GLS version. This option cannot be combined with any other onstat option.	See “Obtaining Utility Version Information” on page 6-1.
-x	Displays information about transactions	See “ onstat -x command: Print database server transaction information” on page 20-215.
-X	Obtains precise information about the threads that are sharing and waiting for buffers	See “ onstat -X command: Print thread information” on page 20-217.
-z	Sets the profile counts to 0	See “ onstat -z command: Clear statistics” on page 20-219.
<i>infile</i>	Specifies a source file for the onstat command	<p>This file must include a previously stored shared-memory segment that you created with the onstat -o command.</p> <p>For instructions on how to create the <i>infile</i> with onstat -o, see “onstat -o command: Output shared memory contents to a file” on page 20-198.</p> <p>For information about running onstat on the source file, see “Running onstat Commands on a Shared Memory Dump File” on page 20-25.</p>

Interactive Execution

To put the **onstat** utility in interactive mode, use the **-i** option. Interactive mode allows you to enter multiple options, one after the other, without exiting the program. For information on using interactive mode, see “**onstat -i** command: Initiate interactive mode” on page 20-189.

Continuous onstat command Execution

Use the **onstat -r** option combined with other **onstat** options to cause the other options to execute repeatedly at a specified interval. For information, see “**onstat -r** command: Repeatedly print selected statistics” on page 20-204.

onstat command: Equivalent to the onstat -pu command

If you invoke **onstat** without any options, the command is interpreted as **onstat -pu** (the **-p** option and the **-u** option).

Syntax:

▶▶ onstat —————▶▶

onstat - command: Print output header

All **onstat** output includes a header. The **onstat -** command displays only the output header and the value that is returned from this command indicates the database server mode.

Syntax:

▶▶ onstat — - —————▶▶

The header takes the following form:

Version--Mode (Type)--(Checkpnt)--Up Uptime--Sh_mem Kbytes

Version

Is the product name and version number

Mode

Is the current operating mode.

(Type)

If the database server uses High-Availability Data Replication, indicates whether the type is primary or secondary

If the database server is not involved in data replication, this field does not appear. If the type is primary, the value P appears. If the type is secondary, the value S appears.

(Checkpnt)

Is a checkpoint flag

If it is set, the header might display two other fields after the mode if the timing is appropriate:

(CKPT REQ)

Indicates that a user thread has requested a checkpoint

(CKPT INP)

Indicates that a checkpoint is in progress. During the checkpoint,

access is limited to read only. The database server cannot write or update data until the checkpoint ends

Uptime

Indicates how long the database server has been running

If the system time is manually changed to the past and the server startup time is later than the current system time, the uptime is not available. In this situation, the header displays the text *Uptime Unavailable*.

Sh_mem

Is the size of database server shared memory, expressed in kilobytes

A sample header for the database server follows:

Informix Version 11.70.UC1--On-Line--Up 15:11:41--9216 Kbytes

Return codes

When you exit the **onstat** utility, there are several useful codes that are displayed. See “Return codes on exiting the **onstat** utility” on page 20-219.

Subheader printed with the onstat - command

If the database server is blocked, the **onstat** header output includes an extra line.

The extra line in the **onstat** header reads as follows:

Blocked: *reason(s)*

The variable *reason* can be one or more of the following values.

Reason	Description
ADMINISTRATION	Database is in administration mode
ARCHIVE	Ongoing storage-space backup
ARCHIVE_EBR	Blocked for External Backup and Recovery.
CHG_PLOG	Blocked while physical log is being changed.
CKPT	Checkpoint
CKPT INP	Interval checkpoint in progress
DBS_DROP	Dropping a dbspace
DDR	Discrete data replication
DYNAMIC_LOG	Log file is being added dynamically
DYNAMIC_LOG_FOR_ER	Log file is being added dynamically in ER setup
FREE_LOG	Log file is being freed
HA_CONV_STD	Blocked while High Availability server is being converted to standard server.
HA_FAILOVER	Blocked while High Availability server failover being processed.
HANG_SYSTEM	Database server failure
LAST_LOG_RESERVED4BACKUP	Waiting for last available log to be backed up
LBU	Logs full high-watermark
LOG_DROP	Log file is being dropped

Reason	Description
LONGTX	Long transaction
MEDIA_FAILURE	Media failure
OVERRIDE_DOWN_SPACE	Waiting to override down dbspace setting because the ONDBSPACEDOWN onconfig parameter is set to WAIT

onstat – command: Print onstat options and functions

Use the **onstat** – command to display a listing of all of the **onstat** options and their functions. You cannot combine this option with any other flag.

Syntax:

```
▶▶ onstat – -- _____ ▶▶
```

Running onstat Commands on a Shared Memory Dump File

You can run **onstat** commands against a shared memory dump file. The shared memory dump file can be produced explicitly by using the **onstat -o** command. If the DUMPSHMEM configuration parameter is set to 1 or set to 2, the dump file is created automatically at the time of an assertion failure.

Syntax:

```
▶▶ onstat –options–infile_____ ▶▶
```

When using the command line, enter the source file as the final argument. The following example prints information about all threads for the shared memory dump contained in the file named `onstat.out`, rather than attempting to attach to the shared memory of a running server.

```
onstat -g ath onstat.out
```

For instructions on how to create the memory dump file with **onstat -o**, see “**onstat -o** command: Output shared memory contents to a file” on page 20-198.

Running onstat Commands on a Shared Memory Dump File Interactively

Use **onstat -i** (interactive mode) to run more than one **onstat** command against a dump file. Interactive mode can save time because the file is read only once. In command-line mode, each command reads the file.

The following example reads the shared memory dump file and enters interactive mode. Other **onstat** commands can be executed against the dump file in the normal interactive fashion.

```
onstat -i source_file
```

For information about interactive mode, see “**onstat -i** command: Initiate interactive mode” on page 20-189.

Running onstat Commands on a Shared Memory Dump File Created Without a Buffer Pool

Certain **onstat** commands have different output when you run them on a dump file created without the buffer pool (created with **onstat -o nobuffs** or with the DUMPSHMEM configuration parameter set to 2):

- If you run **onstat -B** on a dump file created without the buffer pool, the output will display 0 in the memaddr, nslots, and pgflgs columns.
- If you run **onstat -g seg** on a dump file created without the buffer pool, the output will show both the original and nobuffs resident segment size.
- If you run **onstat -P** on a shared-memory dump file that does not have the buffer pool, the output is:

Nobuffs dumpfile -- this information is not available

onstat -a command: Print overall status of the database server

Use the **onstat -a** command to display information about the status of the database server. This command does not display information about all of the onstat options, only about those onstat options used for initial troubleshooting.

Syntax:

►► onstat — -a ————— ◄◄

onstat -b command: Print buffer information for buffers in use

Use the **onstat -b** option to display information about the buffers that are currently in use, including the total number of resident pages in the buffer pool.

Syntax:

►► onstat — -b ————— ◄◄

The maximum number of buffers available is specified in the **buffers** field in the BUFFERPOOL configuration parameter in the ONCONFIG file.

The **onstat -b** command also provides summary information about the number of modified buffers, the total number of resident pages in the buffer pool, the total number of buffers available, the number of hash buckets available, and the size of the buffer in bytes (the page size).

123 modified, 23 resident, 2000 total, 2048 hash buckets, 2048 buffer size.

For information about displaying information about all buffers, use “**onstat -B** command: Prints information about used buffers” on page 20-27.

Example output

Following is sample output from the **onstat -b** command. For a description of the output, see “**onstat -B** command: Prints information about used buffers” on page 20-27.

Buffer pool page size: 4096

address	userthread	flgs	pagenum	memaddr	nslots	pgflgs	xflgs	owner	waitlist
70000001097e9e8	0	c07	1:47841	7000000118e0000	10	1	0	0	0
700000010982188	0	807	1:47827	700000011939000	225	90	10	0	0

2011 modified, 50000 total, 65536 hash buckets, 4096 buffer size

Figure 20-1. **onstat -b** command output

onstat -B command: Prints information about used buffers

Use the **onstat -B** option to display information about buffers that are not on the free-list.

Syntax:

►► onstat — -B ◀◀

Both **onstat -B** and **onstat -b** display the similar information, except that the **onstat -b** command only displays buffers that are currently being accessed by a user thread. The **onstat -B** command displays information for all the buffers that are not on the free-list.

For information about running the **onstat -B** command on a dump file created without the buffer pool, see “Running **onstat** Commands on a Shared Memory Dump File” on page 20-25.

Example output

Buffer pool page size: 4096

address	userthread	flgs	pagenum	memaddr	nslots	pgflgs	xflgs	owner	waitlist
700000010932fe8	0	806	1:40264	700000011150000	5	8b0	0	0	0
700000010933088	0	806	1:40284	700000011151000	5	870	0	0	0
.									
700000010933e48	0	806	1:44585	700000011167000	113	8890	0	0	0
700000010933ee8	0	806	1:42578	700000011168000	5	802	0	0	0
700000010933f88	0	86	1:44348	700000011169000	39	890	0	ffffffffffffffff	0
700000010934028	0	6	1:8	70000001116a000	0	1800	0	0	0
7000000109340c8	0	806	1:246	70000001116b000	5	802	0	0	0
.									
700000010941c28	0	807	1:46486	7000000112ca000	123	801	0	0	0
700000010941ea8	0	806	1:47101	7000000112ce000	10	801	0	0	0

25 modified, 50000 total, 65536 hash buckets, 4096 buffer size

Figure 20-2. **onstat -B** command output

Output description

Buffer pool page size

the size of the buffer pool pages in bytes

address the address of the buffer header in the buffer table

userthread

the address of the most recent user thread to access the buffer table. Many user threads might be reading the same buffer concurrently.

flgs

Uses the following flag bits to describe the buffer:

0x01 Modified data

0x02 Data

0x04 LRU

0x08 Error

pagenum

the physical page number on the disk

memaddr

the buffer memory address

nslots

the number of slot-table entries in the page

This field indicates the number of rows (or portions of a row) that are stored on the page.

pgflgs

Uses the following values, alone or in combination, to describe the page type:

1 Data page

2 Tblspace page

4 Free-list page

8 Chunk free-list page

9 Remainder data page

b Partition resident blobpage

c Blobspace resident blobpage

d Blob chunk free-list bit page

e Blob chunk blob map page

10 B-tree node page

20 B-tree root-node page

40 B-tree branch-node page

80 B-tree leaf-node page

100 Logical-log page

200 Last page of logical log

400 Sync page of logical log

800 Physical log

1000 Reserved root page

2000 No physical log required

8000 B-tree leaf with default flags

xflgs

Uses the following flag bits to describe buffer access:

0x10 share lock

0x80 exclusive lock

owner the user thread that set the **xflgs** buffer flag

waitlist
 the address of the first user thread that is waiting for access to this buffer

For a complete list of all threads waiting for the buffer, refer to “**onstat -X** command: Print thread information” on page 20-217.

onstat -c command: Print ONCONFIG file contents

Use the **onstat -c** command to display the contents of the ONCONFIG file.

Syntax:

```
▶▶ onstat -c ◀◀
```

The database server first checks if you have assigned a value to the environment variable **ONCONFIG**. You can use the **onstat -c** option with the database server in any mode, including offline.

UNIX Only:

On UNIX, if you have set **ONCONFIG**, **onstat -c** displays the contents of the **\$INFORMIXDIR/etc/\$ONCONFIG** file. If not, by default, **onstat -c** displays the contents of **\$INFORMIXDIR/etc/onconfig**.

Windows Only:

On Windows, if you have set **ONCONFIG**, **onstat -c** displays the contents of the **%INFORMIXDIR%\etc\%ONCONFIG%** file. If not, by default, **onstat -c** displays the contents of **%INFORMIXDIR%\etc\onconfig**.

onstat -C command: Print B-tree scanner information

Use the **-C** command to display information about the B-tree scanner subsystem and each B-tree scanner thread.

Syntax:

```
▶▶ onstat -C
    prof
    hot
    part
    clean
    range
    map
    alice
    all
    ◀◀
```

The following options are available with the **onstat -C** command and can be combined:

prof Prints the profile information for the system and each B-tree scanner thread. This is the default option.

hot Prints the hot list index key in the order to be cleaned

- part* Prints all partitions with index statistics
- clean* Prints information about all the partitions that were cleaned or need to be cleaned
- range* Prints the savings in pages processed by using index range scanning
- map** Displays the current bitmaps for each index being cleaned by the alice cleaning method
- alice** Displays the efficiency of the alice cleaning method option
- all* Prints all **onstat -C** options

Example output using the prof option

```

Btree Cleaner Info
BT scanner profile Information
=====
Active Threads                      1
Global Commands                    2000000 Building hot list
Number of partition scans          11003
Main Block                         0xc000000003c9dc68
BTC Admin                          0xc0000000024bc208

BTS info      id  Prio  Partnum      Key      Cmd
0xc000000003c9dee8  0  High  0x00000000      0      40 Yield N
  Number of leaves pages scanned      77
  Number of leaves with deleted items    6
  Time spent cleaning (sec)            0
  Number of index compresses           0
  Number of deleted items             113
  Number of index range scans          0
  Number of index leaf scans           0
  Number of index alice scans          2

```

Figure 20-3. **onstat -C** command output with the *prof* option

Output description using the prof option

- Id* BTSCANNER ID
- Prio* Current priority of BTSCANNER
- Partnum*
The partition number for the index this thread is currently working on
- Cmd** Command this thread is processing currently

Example output using the hot option

Btree Cleaner Info				
Index Hot List				
=====				
Current Item	5	List Created	15:29:47	
List Size	4	List expires in	0 sec	
Hit Threshold	500	Range Scan Threshold	-1	
Partnum	Key	Hits		
0x00100191	1	14 *		
0x00A00022	1	13 *		
0x00100191	2	8 *		
0x00100150	2	7 *		

Figure 20-4. **onstat -C** command output with the hot option

Output description using the hot option

Partnum

The partition number for an index

Key Index Key

Hits The current value of the Hit counter

* Indicates that this partition has been cleaned during this hot list duration

Example output using the part option

Btree Cleaner Info				
Index Statistics				
=====				
Partnum	Key	Positions	Compress	Split
0x00100002	1	146	0	0
0x00100004	1	4	0	0
0x00100004	2	13	0	0
0x00100005	1	1	0	0
0x00100005	2	0	0	0
0x00100006	1	1	0	0
0x00100006	2	0	0	0
0x00100007	2	1	0	0
0x00100008	2	1	0	0
0x0010000a	1	0	0	0
0x0010000e	3	1	0	0
0x00100011	1	1	0	0
0x00100013	2	2	0	0

Figure 20-5. **onstat -C** command output with the part option

Output description using the part option

Partnum

The partition number for an index

Key Index Key

Positions

Number of times index has been read

Compress

Number of pages which have been compressed

Split

Number of splits that have occurred

C

Indicates partition is busy being cleaned

N

Index partition no longer eligible for cleaning

Example output using the clean option

Btree Cleaner Info								
Index Cleaned Statistics								
=====								
Partnum	Key	Dirty	Hits	Clean Time	Pg Examined	Items Del	Pages/Sec	
0x00100013	2		2	0	0	0	0.00	
0x0010008b	3		1	0	0	0	0.00	
0x001000c7	1		2	0	0	0	0.00	
0x00100150	2		7	0	0	0	0.00	
0x0010016f	2		2	0	0	0	0.00	
0x00100191	1		14	0	0	0	0.00	
0x00100191	2		8	0	0	0	0.00	
0x00a00011	2		6	0	0	0	0.00	
0x00a00013	1		0	0	24	0	24.00	
0x00a00019	1		0	0	470	225	470.00	
0x00a00022	1		13	0	0	0	0.00	
0x00a00022	2		5	0	0	0	0.00	

Figure 20-6. **onstat -C** command output with the clean option

Output description using the clean option

Partnum

The partition number for an index

Key

Index Key

Dirty Hits

Number of times a dirty page has been scanned

Clean Time

Total time spent, in seconds

Pg Examined

Number of pages examined by btscanner thread

Items Del

Number of items removed from this index

Pages/Sec

Number of pages examined per second

C

Indicates partition is busy being cleaned

N

index partition is no longer eligible for cleaning

Example Output

Btree Cleaner Info					
Cleaning Range Statistics					
=====					
Partnum	Key	Low	High	Size	Saving
0x001001bc	2	36	69	96	65.6 %
0x001001be	1	16	20	48	91.7 %
0x001001cd	1	8	21	32	59.4 %
0x001001cd	2	24	25	32	96.9 %

Figure 20-7. **onstat -C** range

Output Description

Partnum

The partition number

Key

Index Key

Low

Low boundary for range scan

High

High boundary for index scan

Size

Size of index in pages

Saving

Percentage of time saved versus a full scan

C

Indicates partition is busy being cleaned

N

Index partition is no longer eligible for cleaning

Example Output

Btree Cleaner Info		
ALICE Bitmap of Deleted Index Items		
=====		
Partnum	Key	Map
0x00100013	2 0000:	80000000 00000000
0x0010008b	3 0000:	80000000 00000000
0x001000c7	1 0000:	80000000 00000000
0x00100150	2 0000:	80000000 00000000
0x0010016f	2 0000:	80000000 00000000
0x00100191	1 0000:	80000000 00000000
0x00100191	2 0000:	80000000 00000000
0x00a00011	2 0000:	80000000 00000000
0x00a00013	1 0000:	00000000 00000000
0x00a00019	1 0000:	00000000 00000000
0x00a00022	1 0000:	80000000 00000000
0x00a00022	2 0000:	80000000 00000000

Figure 20-8. **onstat -C** map

Output Description

Partnum

The partition number

Key

Index Key

Map

Alice bitmap

Example Output

```
Btree Cleaner Info

ALICE Cleaning Statistics
=====

System ALICE Info: Mode =    6, Eff =    30 %, Adj =    5

Partnum  Mode BM_Sz  Used_Pg  Examined  Dirty_Pg  # I/O  Found   Eff   Adj
0x00100013  6    64    97        0         0      0      0    0.0 %    0
0x0010008b  6    64     5        0         0      0      0    0.0 %    0
0x001000c7  6    64     2        0         0      0      0    0.0 %    0
0x00100150  6    64    91        0         0      0      0    0.0 %    0
0x0010016f  6    64    91        0         0      0      0    0.0 %    0
0x00100191  6    64    26        0         0      0      0    0.0 %    0
0x00100191  6    64    26        0         0      0      0    0.0 %    0
0x001001bc  0     0    91        0         0      0      0    0.0 %    0
0x001001cd  0     0    26        0         0      0      0    0.0 %    0
0x001001cd  0     0    26        0         0      0      0    0.0 %    0
0x00a00011  6    64    91        0         0      0      0    0.0 %    0
0x00a00013  6    64    25        24         3      3      1   33.3 %    1
0x00a00019  6    64   470       470         3      3      2   66.7 %    1
0x00a00022  6    64    26        0         0      0      0    0.0 %    0
0x00a00022  6    64    26        0         0      0      0    0.0 %    0
```

Figure 20-9. **onstat -C** *alice*

Output Description

Partnum

The partition number for an index

Mode

The *alice* mode for the current partition

BM_Sz

The size allocated for the bitmap

Used_Pg

The size of the index in pages (used)

Dirty_Pg

Number of dirty pages

I/O

Number of pages read

Found

Number of dirty pages found in reads

Eff

How efficient was the bitmap

Adj

Number of times the *alice* efficiency level for the partition was insufficient and was adjusted

onstat -d command: Print chunk information

Use the **onstat -d** command to display information about chunks in each storage space.

Syntax:

►► onstat — -d ————— ►►

Using onstat -d with sbspaces

For information about using **onstat -d** to determine the size of sbspaces, user-data areas, and metadata areas, see monitoring sbspaces in the *IBM Informix Administrator's Guide*.

Using onstat -d with blobspaces

If you issue the **onstat -d** command on an instance with blobspace chunks, the database server displays the following message:

NOTE: For BLOB chunks, the number of free pages shown is out of date.
Run 'onstat -d update' for current stats.

To obtain the current statistics for blobspace chunks, issue the **onstat -d update** command. The **onstat** utility updates shared memory with an accurate count of free pages for each blobspace chunk. The database server displays the following message:

Waiting for server to update BLOB chunk statistics ...

Example output

You can interpret output from this option as follows.

```
BM Informix Dynamic Server Version 11.70.F          -- On-Line -- Up 00:01:27 -- 133540 Kbytes

Dbspaces
address  number  flags    fchunk  nchunks  pgsize  flags  owner   name
48750028 1       0x60001  1       1        2048   N BA   informix rootdbs
4a0bee00 2       0x60001  2       1        2048   N BA   informix dbspace2
  2 active, 2047 maximum

Chunks
address  chunk/dbs  offset  size    free    bpages  flags  pathname
487501c8 1          1      0      1000000 923615   PO-B-- /dev/raw/raw1
49f1bda0 2          2      0       5000    4972    PO-BED /work2/dbspaces/dbs2
  2 active, 32766 maximum

Chunks
address  chunk/dbs  offset  size    free    bpages  flags  pathname
460bb1c0 1          1      0       60000   24387   PO-B-D /work8/root_chunk
460bb7a0 2          3      0       2500    2447    PO-BED /work8/dbspaces/cook1
460bb990 3          4      0       2500    2447    PO-BED /work8/dbspaces/cook2
460bbb80 4          5      0      10000   9947    PO-BE- /work8/dbspaces/dbtmp
460bbd70 5          6      0     100000 30483    PO-BED /work8/dbspaces/dbs1
  5 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.

Expanded chunk capacity mode: always
```

Figure 20-10. **onstat -d** command output

Output description - Dbspaces

The first section of the display describes the storage spaces:

address Is the address of the storage space in the shared-memory space table

number

Is the unique ID number of the storage space assigned at creation

flags

Uses hexadecimal values to describe each storage space. The individual flag values can be summed to show cumulative properties of the dbspace. The following table describes each hexadecimal value:

Table 20-21. Descriptions for each hexadecimal value

Flag Value	Description
0x00000000	Mirror not allowed and dbspace is unmirrored
0x00000001	Mirror is allowed and dbspace is unmirrored
0x00000002	Mirror is allowed and dbspace is mirrored
0x00000004	Down
0x00000008	Newly mirrored
0x00000010	Blobspace
0x00000020	Blobspace on removable media
0x00000040	Blobspace is on optical media
0x00000080	Blobspace is dropped
0x00000100	Blobspace is the optical STAGEBLOB
0x00000200	Space is being recovered
0x00000400	Space is fully recovered
0x00000800	Logical log is being recovered
0x00001000	Table in dbspace is dropped
0x00002000	Temporary dbspace
0x00004000	Blobspace is being backed up
0x00008000	Sbospace
0x00010000	Physical or logical log changed
0x00020000	Dbspace or chunk tables have changed
0x00040000	Dbspace or blobspace contains large chunk
0x00080000	Chunk in this dbspace has been renamed
0x00100000	Temporary dbspace used by only by shared disk secondary server. It is one of the dbspaces listed in the SDS_TEMPDBS configuration parameter on the SD secondary server.
0x00200000	Temporary dbspace for the SD secondary server. This is listed in the DBSPACETEMP configuration parameter on the shared disk secondary server.
0x00400000	The dbspace was externally backed up.

fchunk The ID number of the first chunk

nchunks

The number of chunks in the storage space

pgsize

The size of the dbspace pages in bytes

flags

Uses the following letter codes to describe each storage space:

Position 1:

Flag	Description
M	Mirrored
N	Not mirrored

Position 2:

Flag	Description
X	Newly mirrored
P	Physically recovered, waiting for logical recovery
L	Being logically recovered
R	Being recovered
D	Down

Position 3:

Flag	Description
B	Blobspace
S	Sbpace
T	Temporary dbspace
U	Temporary sbpace
W	Temporary dbspace on primary server (This flag is shown on SD secondary servers only.)

Position 4:

Flag	Description
B	The dbspace can have large chunks that are greater than 2 GB.

Position 5:

Flag	Description
A	The dbspace is auto-expandable, because the SP_AUTOEXPAND configuration parameter is enabled and the dbspace is configured with a create size or extend size that is not zero.

owner The owner of the storage space

name The name of the storage space

In the line immediately following the storage-space list, **active** refers to the current number of storage spaces in the database server instance including the rootdbs and **maximum** refers to total *allowable* spaces for this database server instance.

Output description - Chunks

The second section of the **onstat -d** command output describes the chunks:

address

The address of the chunk

chk/db

The chunk number and the associated space number

offset The offset into the file or raw device in base page size

size The size of the chunk in terms of the page size of the dbspace to which it belongs.

free The number of unallocated pages in the chunk in units of the page size of the associated dbspace. A value of 0 indicates that all the space in the chunk has been allocated to tables, but does not indicate how much space is free inside the tables. For example, suppose you create a dbspace with one chunk of 200 MB and create one table with an extent size of 200 MB. The value of the **free** field is 0, indicating that the chunk has no free space, however, the new empty table has 200 MB of free space.

For a blob space, a tilde indicates an approximate number of unallocated blob pages.

For an sbspace, indicates the number of unallocated pages of user data space and total user data space.

bpages

Is the size of the chunk in blob pages

Blob pages can be larger than disk pages; therefore, the **bpages** value can be less than the **size** value.

For an sbspace, is the size of the chunk in sbpages.

flags Provides the chunk status information as follows:

Position 1:

Flag	Description
P	Primary
M	Mirror

Position 2:

Flag	Description
N	Renamed and either Down or Inconsistent
O	Online
D	Down
X	Newly mirrored
I	Inconsistent

Position 3:

Flag	Description
-	Dbspace
B	Blob space
S	Sbspace

Position 4:

Flag	Description
B	The dbspace can have large chunks that are greater than 2 GB.

Position 5:

Flag	Description
E	Identifies the chunk as extendable
-	Identifies the chunk as not extendable

Position 6:

Flag	Description
-	Not using the direct I/O or concurrent I/O option for this cooked file chunk
C	On AIX, using the concurrent I/O option for this cooked file chunk
D	Using the direct I/O option for this cooked file chunk

pathname

The pathname of the physical device

In the line immediately following the chunk list, **active** displays the number of active chunks (including the root chunk) and **maximum** displays the total number of chunks.

For information about page reads and page writes, refer to “**onstat -D** command: Print page-read and page-write information.”

Related reference:

“modify chunk extend argument: Extend the size of a chunk (SQL administration API)” on page 21-65

onstat -D command: Print page-read and page-write information

Use the **onstat -D** command to display page-read and page-write information for the first 50 chunks in each space.

Syntax:

►► onstat — -D ————— ►►

Example output

```

Dbspaces
address  number  flags      fchunk  nchunks  pgsz  flags  owner  name
a40d7d8  1             0x1      1        1      2048  N      informix rootdbs
1 active, 2047 maximum

Chunks
address  chunk/dbs  offset  page Rd  page Wr  pathname
a40d928  1          1      0        0        0      /work/11.1/dbspaces/stardbs3
1 active, 2047 maximum

Expanded chunk capacity mode: disabled

```

Figure 20-11. **onstat -D** command output

Output description

The output of **onstat -D** is almost identical to the output of **onstat -d**. The following columns are unique to **onstat -D**. For information on the other output columns see “**onstat -d** command: Print chunk information” on page 20-34.

page Rd

Is the number of pages read

page Wr

Is the number of pages written

onstat -f command: Print dbspace information affected by dataskip

Use the **-f** command to list the dbspaces that the dataskip feature currently affects.

Syntax:

```

▶▶ onstat -f ◀◀

```

The **-f** option lists both the dbspaces that were set with the DATASKIP configuration parameter and the **-f** option of **onspaces**. When you execute **onstat -f**, the database server displays one of the following three outputs:

- Dataskip is OFF for all dbspaces.
- Dataskip is ON for all dbspaces.
- Dataskip is ON for the following dbspaces:
dbspace1 dbspace2...

onstat -F command: Print counts

Use the **onstat -F** command to display a count for each type of write that flushes pages to disk.

Syntax:

```

▶▶ onstat -F ◀◀

```

Example output

Fg Writes		LRU Writes		Chunk Writes			
0		330		7631			
address	flusher	state	data	# LRU	Chunk	Wakeup	Idle Time
c7c8850	0	I	0	9	29	16116	16093.557
states: Exit Idle Chunk Lru							

Figure 20-12. **onstat -F** command output

Output description

You can interpret output from this option as follows:

Fg Writes

Is the number of times that a foreground write occurred

LRU Writes

Is the number of times that an LRU write occurred

Chunk Writes

Is the number of times that a chunk write occurred

address Is the address of the user structure assigned to this page-cleaner thread

flusher Is the page-cleaner number

state Uses the following codes to indicate the current page-cleaner activity:

- C** Chunk write
- E** Exit
- I** Cleaner is idle
- L** LRU queue

The exit code indicates either that the database server is performing a shutdown or that a page cleaner did not return from its write in a specific amount of time. When an operation fails to complete within the allotted time, this situation is known as a time-out condition. The database server does not know what happened to the cleaner, so it is marked as **exit**. In either case, the cleaner thread eventually exits.

data Provides additional information in concert with the **state** field

If **state** is **C**, **data** is the chunk number to which the page cleaner is writing buffers. If **state** is **L**, **data** is the LRU queue from which the page cleaner is writing. The **data** value is displayed as a decimal, followed by an equal sign, and repeated as a hexadecimal.

#LRU Corresponds to the **onstat -g ath** thread ID output

Chunk

Number of chunks cleaned

Wakeup

Number of times the flusher thread was awoken

Idle Time

Time in seconds the flusher thread has been idle

onstat -g monitoring options

The options that you can use with **onstat -g** command are used for support and debugging only. You can include only one of these options in the **onstat -g** command.

The following table lists the **onstat -g** options. For more information, see the *IBM Informix Performance Guide*.

onstat -g Option	Topic or Function
-g act	Prints active threads. See “ onstat -g act command: Print active threads” on page 20-48.
-g afr	Prints allocated memory fragments for a specified session or shared-memory pool. Each session is allocated a pool of shared memory. To obtain the pool name, see the -mem option. See “ onstat -g afr command: Print allocated memory fragments” on page 20-48.
-g all	Prints output from all onstat -g options. See “ onstat -g all command: Print diagnostic information” on page 20-49.
-g aqt	Prints information about the data marts and the associated accelerated query tables (AQTs). See “ onstat -g aqt command: Print data mart and accelerated query table information” on page 20-49.
-g ath	Prints all threads. The sqlmain threads represent client sessions. The rstcb value corresponds to the user field of the onstat -u command. See “ onstat -g ath command: Print information about all threads” on page 20-54. For information about using onstat -g ath to print Enterprise Replication threads, see the <i>IBM Informix Enterprise Replication Guide</i> .
-g buf	Prints profile information for each buffer pool. See “ onstat -g buf command: Print buffer pool profile information” on page 20-55.
-g cat	Prints information from the Enterprise Replication global catalog. The global catalog contains a summary of information about the defined servers, replicates, and replicate sets on each of the servers within the enterprise. See “ onstat -g cat command: Print ER global catalog information” on page 20-57.
-g cac agg	Prints the definitions for user-defined aggregates that are currently in the cache.
-g cac stmt	Prints the contents of the SQL statement cache. Prints the same output as the -g ssc option.
-g cdr	Prints the settings of Enterprise Replication configuration parameters and environment variables. See “ onstat -g cdr config command: Print ER settings” on page 20-59.
-g ckp	Prints the checkpoint history and displays configuration recommendations if a suboptimal configuration is detected. See “ onstat -g ckp command: Print checkpoint history and configuration recommendations” on page 20-61.
-g cluster	Prints high-availability cluster information. See “ onstat -g cluster command: Print high-availability cluster information” on page 20-65.
-g cmsm	Prints Connection Manager daemon instances and displays the number of connections each daemon has processed. See “ onstat -g cmsm command: Print Connection Manager information” on page 20-69.
-g con	Prints conditions with waiters. See “ onstat -g con command: Print condition and thread information” on page 20-71.

onstat -g Option	Topic or Function
-g cpu	Prints information about runtime statistics for all the threads running in the server. See “ onstat -g cpu : Print runtime statistics” on page 20-71.
-g dbc	Prints information for database scheduler and the SQL administration API. See “ onstat -g dbc command: Print dbScheduler and dbWorker thread statistics” on page 20-73.
-g ddr	Prints the status of the Enterprise Replication database log reader. If log reading is blocked, data might not be replicated until the problem is resolved. See “ onstat -g ddr command: Print ER database log reader status” on page 20-74.
-g dic	Prints one line of information for each table cached in the shared-memory dictionary. If given a specific table name as a parameter, prints internal SQL information for that table. For more information, see your <i>IBM Informix Performance Guide</i> . For sample output, see “ onstat -g dic command: Print table information” on page 20-76.
-g dis	Prints a list of database servers and their status, and information about each database server, INFORMIXDIR , sqlhosts file, ONCONFIG file, and hostname. See “ onstat -g dis command: Print database server information” on page 20-77.
-g dll	Prints a list of dynamic libraries that have been loaded. See “ onstat -g dis command: Print database server information” on page 20-77.
-g dmp	Prints raw memory at a given address for a number of given bytes. See “ onstat -g dmp command: Print raw memory” on page 20-80.
-g dri	Prints data-replication information. See monitoring High-Availability Data-Replication status in the <i>IBM Informix Administrator's Guide</i> . See “ onstat -g dri command: Print high-availability data replication information” on page 20-81.
-g dsc	Prints data-distribution cache information. See “ onstat -g dsc command: Print distribution cache information” on page 20-83.
-g dss	Prints detailed statistical information about the activity of individual data sync threads and about user-defined data types. See “ onstat -g dss command: Print ER environment data” on page 20-84.
-g dsk	Prints information that shows the progress of currently running compression operations, such as compress, repack, and uncompress. See “ onstat -g dsk command: Print the progress of the currently running compression operation” on page 20-78.
-g dtc	Prints statistics about the delete table cleaner which removes rows from the delete table when they are no longer needed. See “ onstat -g dtc command: Print delete table cleaner statistics” on page 20-85.
-g env	Prints the values of environment variables the database server currently uses. See “ onstat -g env command: Print environment variable values” on page 20-86.
-g ffr	Prints free fragments for a pool of shared memory. See “ onstat -g ffr command: Print free fragments” on page 20-88.
-g glo	Prints global multithreading information. This information includes CPU use information about the virtual processors, the total number of sessions, and other multithreading global counters. On Windows, the virtual processors are operating system threads. The values displayed under the pid field are thread IDs not process IDs. (Windows). See “ onstat -g glo command: Print global multithreading information” on page 20-89.

onstat -g Option	Topic or Function
-g grp	Prints statistics about the Enterprise Replication grouper. The grouper evaluates the log records, rebuilds the individual log records into the original transaction, packages the transaction, and queues the transaction for transmission. See “ onstat -g grp command: Print ER grouper statistics” on page 20-91.
-g his	Prints information about the SQLTrace configuration parameter. See “ onstat -g his command: Print SQL trace information” on page 20-97.
-g imc	Prints information about Informix MaxConnect instances that are connected to the database server. If Informix MaxConnect is not connected to the database server, this command displays No MaxConnect servers are connected.
-g ioa	Prints combined information from the -g ioq and -g iov command. See “ onstat -g ioa command: Print combined onstat -g information” on page 20-102.
-g iob	Prints the big buffer usage summary. See “ onstat -g iob command: Print big buffer use summary” on page 20-103.
-g iof	Prints asynchronous I/O statistics by chunk or file. This option is similar to the -D option, except it also displays information about nonchunk, temporary, and sort-work files. See “ onstat -g iof command: Print asynchronous I/O statistics” on page 20-104.
-g iog	Prints AIO global information. See “ onstat -g iog command: Print AIO global information” on page 20-105.
-g ioq	Prints pending I/O operations for the <i>queue name</i> . If given the <i>gfd</i> or <i>kaio</i> queue name, a queue for each CPU VP is displayed. If <i>queue name</i> is omitted, I/O statistics for all queues are displayed. See “ onstat -g ioq command: Print I/O queue information” on page 20-105.
-g iov	Prints asynchronous I/O statistics by virtual processor. See “ onstat -g iov command: Print AIO VP statistics” on page 20-107.
-g ipl	Prints index page logging status. See “ onstat -g ipl command: Print index page logging status information” on page 20-107.
-g lap	Prints information about the status of light appends. See “ onstat -g lap command: Print light appends status information” on page 20-109.
-g laq	Prints information about queues on the secondary server. See “ onstat -g laq command: Print secondary server queues” on page 20-109.
-g lmm	Prints information about automatic low memory management settings and recent activity. See “ onstat -g lmm command: Print low memory management information” on page 20-111.
-g lmx	Prints all locked mutexes. See “ onstat -g lmx command: Print all locked mutexes” on page 20-113.

onstat -g Option	Topic or Function
-g mem	<p>Prints statistics for a memory pool. Also prints the pool name, type of shared memory segment that contains the pool, the address of the pool, the total size of the pool, the number of bytes of free memory that it contains, and the number of free and allocated fragments in the pool. If no argument is provided, displays information about all pools. The block pools are listed in a separate section after the main pool list.</p> <p>You also can use ISA to obtain detailed information about a memory pool. If you run an SQL query that allocates memory from the PER_STMT_EXEC and PER_STMT_PREP memory duration pools, the onstat -g mem command displays information about the PRP.sessionid.threadid pool and the EXE.sessionid.threadid pool.</p> <p>See “onstat -g mem command: Print pool memory statistics” on page 20-114. For more information, see the <i>IBM Informix DataBlade API Programmer's Guide</i>.</p>
-g mgm	Prints Memory Grant Manager resource information. See “ onstat -g mgm command: Print MGM resource information” on page 20-116.
-g nbm	Prints block bit map for the nonresident segments, one bit per 8-kilobyte block. Bit set indicates block free. See “ onstat -g nbm command: Print a block bit map” on page 20-119.
-g nif	Prints statistics about the network interface. This command is useful to determine why data is not replicating. See “ onstat -g nif command: Print statistics about the network interface” on page 20-120.
-g nsc	Prints shared-memory status by <i>client id</i> . If <i>client id</i> is omitted, all client status areas are displayed. This command prints the same status data as the nss option. See “ onstat -g nsc command: Print current shared memory connection information” on page 20-121.
-g nsd	Prints network shared-memory data for poll threads. See “ onstat -g nsd command: Print poll threads shared-memory data” on page 20-124.
-g nss	Prints network shared-memory status by <i>session id</i> . If <i>session id</i> is omitted, all session status areas are displayed. This command prints the same status data as the nsc option. See “ onstat -g nss command: Print shared memory network connections status” on page 20-124.
-g nta	Prints combined network statistics from the -g ntd , -g ntm , -g ntt , and -g ntui commands. If Informix MaxConnect is installed, this command prints statistics that you can use to tune Informix MaxConnect performance.
-g ntd	Prints network statistics by service. See “ onstat -g ntd command: Print network statistics” on page 20-125.
-g ntm	Prints network mail statistics. See “ onstat -g ntm command: Print network mail statistics” on page 20-126.
-g ntt	Prints network user times. See “ onstat -g ntt command: Print network user times” on page 20-126.
-g ntui	Prints network user statistics. See “ onstat -g ntui command: Print network user statistics” on page 20-127.
-g opn	Prints open partitions. See “ onstat -g opn command: Print open partitions” on page 20-127.

onstat -g Option	Topic or Function
-g osi	Prints information about operating system resources and parameters. See “ onstat -g osi : Print operating system information” on page 20-129.
-g pos	Prints \$INFORMIXDIR/etc/ .infos.DBSERVERNAME file for UNIX or the %INFORMIXDIR%\etc\ .infos.DBSERVERNAME file for Windows. See “ onstat -g pos command: Print file values” on page 20-130.
-g ppd	Prints information about compressed fragments (also referred to as partitions) and the compression dictionary. See “ onstat -g ppd command: Print partition compression dictionary information” on page 20-131.
-g ppf	Prints partition profile for the specified partition number or prints profiles for all partitions. See “ onstat -g ppf command: Print partition profiles” on page 20-132.
-g pqs	Prints SQL statement information. You can display information for all sessions or for a specific <i>sessionid</i> . See “ onstat -g pqs command: Print operators for all SQL queries” on page 20-133.
-g prc	Prints information about SPL routine cache. See “ onstat -g prc command: Print sessions using UDR or SPL routine” on page 20-134.
-g proxy	Prints proxy distributor information. See “ onstat -g proxy command: Print proxy distributor information” on page 20-135.
-g qst	Prints queue wait statistics. See “ onstat -g qst command: Print wait options for mutex and condition queues” on page 20-141.
-g que	Prints statistics for the high-level queue interface (which are common to all the queues of the Enterprise Replication Queue Manager. See “ onstat -g que command: Prints ER queue statistics” on page 20-140.
-g rbm	Prints block bit map for the resident segment (communication message area). See “ onstat -g rbm command: Print a block map of shared memory” on page 20-143.
-g rcv	Prints statistics about the receive manager, which is a set of service routines between the receive queues and data sync. See “ onstat -g rcv command: Print ER receive manager statistics” on page 20-144.
-g rea	Prints ready threads. See “ onstat -g rea command: Print ready threads” on page 20-147.
-g rep	Prints events that are in the queue for the schedule manager. See “ onstat -g rep command: Print ER schedule manager events” on page 20-148.
-g rqm	Prints statistics and contents of the low-level queues (each individual queue) managed by the Reliable Queue Manager (RQM). See “ onstat -g rqm command: Print low-level queue statistics” on page 20-148.
-g rss	Prints remote standalone secondary (RSS) server information. See “ onstat -g rss command: Print RS secondary server information” on page 20-151.
-g rwm	Prints read/write mutexes. See “ onstat -g rwm command: Print read and write mutexes” on page 20-154.
-g sch	Prints the number of semaphore operations, spins, and busy waits for each virtual processor. On Windows, the virtual processors are operating system threads. The values displayed under the pid field are thread IDs not process IDs. See “ onstat -g sch command: Print VP information” on page 20-155.

onstat -g Option	Topic or Function
-g scn	Prints information about scans based on rows scanned. See “ onstat -g scn command: Print scan information” on page 20-155
-g sds	Prints shared disk secondary (SDS) server information. See “ onstat -g sds command: Print SD secondary server information” on page 20-157.
-g seg	Prints shared-memory-segment statistics. This option shows the number and size of shared-memory segments that the database server is currently using. See “ onstat -g seg command: Print shared memory segment statistics” on page 20-161.
-g ses	Prints session information by <i>sessionid</i> . If <i>sessionid</i> is missing, a one-line summary of each session prints. See “ onstat -g ses command: Print session-related information” on page 20-162.
-g sle	Prints all sleeping threads. See “ onstat -g sle command: Print all sleeping threads” on page 20-167.
-g smb	Prints detailed information about sbspaces. See “ onstat -g smb command: Print sbspaces information” on page 20-167.
-g smx	Displays server multiplexer group connections information. See “ onstat -g smx command: Print multiplexer group information” on page 20-169.
-g spi	Prints spin locks with long spins. See “ onstat -g spi command: Print spin locks with long spins” on page 20-170.
-g sql	Prints SQL information by <i>session id</i> . If <i>session id</i> is omitted, a one-line summary for each session prints. See “ onstat -g sql command: Print SQL-related session information” on page 20-172.
-g src	Searches for patterns in shared memory. See “ onstat -g src command: Patterns in shared memory” on page 20-173.
-g ssc	Monitors the number of times that the database server reads the SQL statement in the cache. Optional keywords include all and pool . For the syntax, option descriptions, and example output, see “ onstat -g ssc command: Print SQL statement occurrences” on page 20-174. Displays the same output as onstat -g cac stmt . For more information, see improving query performance in the <i>IBM Informix Performance Guide</i> .
-g stk	Prints the stack of a specified thread or prints stacks for <i>all</i> threads. This option is not supported on all platforms and is not always accurate. See “ onstat -g stk command: Print thread stack” on page 20-176.
-g stm	Prints the memory that each prepared SQL statement uses. See “ onstat -g stm command: Print SQL statement memory usage” on page 20-176. For more information, see memory utilization and improving query performance in the <i>IBM Informix Performance Guide</i> .
-g stq	Prints queue stream information. See “ onstat -g stq command: Print queue information” on page 20-177.
-g sts	Prints maximum and current stack use per thread. See “ onstat -g sts command: Print stack usage for each thread” on page 20-177.
-g sync	Prints which sync is active. See “ onstat -g sync command: Print ER synchronization status” on page 20-179.
-g tpf	Prints thread profile for a specific thread ID. See “ onstat -g tpf command: Print thread profiles” on page 20-180.

onstat -g Option	Topic or Function
-g ufr	Prints allocated fragments by use. See “ onstat -g ufr command: Print memory pool fragments” on page 20-181.
-g vpcache	Prints information about CPU VP memory block cache statistics. See “ onstat -g vpcache command: Print CPU VP memory block cache statistics” on page 20-182.
-g wai	Prints waiting threads; all threads waiting on mutex or condition, or yielding. See “ onstat -g wai command: Print wait queue thread list” on page 20-184.
-g wmx	Prints all mutexes with waiters. See “ onstat -g wmx command: Print all mutexes with waiters” on page 20-184.
-g wst	Prints wait statistics for threads. See “ onstat -g wst command: Print wait statistics for threads” on page 20-185.

onstat -g act command: Print active threads

Use the **onstat -g act** command to display information about the active threads.

Syntax:

```
►► onstat — -g — act — ◀◀
```

Following is sample output from the **onstat -g act** command. For a description of the output, see “**onstat -g ath** command: Print information about all threads” on page 20-54.

Example output

```
Running threads:
tid  tcb      rstcb  prty  status  vp-class  name
2    b3132d8    0      1    running *2adm    adminthd
40   c5384d0    0      1    running *1cpu    tlitcpoll
```

Figure 20-13. **onstat -g act** command output

onstat -g afr command: Print allocated memory fragments

Use the **onstat -g afr** command to display information about the allocated memory fragments for a specified session or shared-memory pool. Each session is allocated a pool of shared memory.

This command requires an additional argument to specify either a pool name or session ID whose memory pool information is to be displayed. Each session is allocated a memory pool with the same name as the session ID. Use the **onstat -g mem** command to identify the pool name and the **onstat -g ses** command to identify the session ID.

Syntax:

```
►► onstat — -g — afr — pool_name  
sessionid — ◀◀
```

Example output

```
Allocations for pool name global:
addr      size    memid    fileid  location
4b231000  3288    overhead  306     mtshpool.c:617
4b231cd8   72      mcbmsg    1637    rldmsg.c:92
4b231d20  160     mcbmsg    1637    rldmsg.c:92
4b231dc0   64      osenv     2909    osenv.c:1164
4b231e00   64      osenv     2909    osenv.c:1971
4b231e40   64      osenv     2909    osenv.c:1164
4b231e80   64      osenv     2909    osenv.c:1971
```

Figure 20-14. **onstat -g afr** command output

Output description

addr (hexadecimal)

Memory address of the pool fragment.

size (decimal)

Size, in bytes, of the pool fragment.

memid (string)

Memory ID of the pool fragment.

fileid (decimal)

Internal use only. Code file identifier for the allocation.

location (string)

Internal use only. Line number in the code for the allocation.

onstat -g all command: Print diagnostic information

Use the **onstat -g all** command to gather diagnostic information if advised to do so by IBM Support. For normal administrative purposes, use the **onstat -g** command with individual options.

Syntax:

```
▶▶ onstat -g all ▶▶
```

onstat -g aqt command: Print data mart and accelerated query table information

Use the **onstat -g aqt** command to display information about the data marts and the associated accelerated query tables (AQTs).

Syntax:

```
▶▶ onstat -g aqt [aqt_name] ▶▶
```

Example output

AQT Dictionary Cache for database school:

mart: school
 accelerator: DWAFINAL
 last load: 2011/07/29 07:00:39

AQT name	FactTab	#tab	#matched	address
aqt4d11b552-7d41-4b0c-824b-7714b6cb580a	103	1	328	0x4d187e08
aqt61498fab-3617-4c8c-ab40-fd8af4253998	103	2	42	0x4d84a448
aqtbcb2da77c-bca8-4ce7-9191-8180a860da34	103	2	768	0x4d187f60
aqt88757e9d-81ee-43b4-87b2-0bf48c98fa55	103	3	15	0x4d84a190
aqta786d0dc-8e95-4de0-a1bd-773aa03a52db	103	3	1475	0x4d84a650
aqt8dd61c80-2c1c-4f0e-8f0c-91babe789f41	103	4	632	0x4d84a908

mart: school2
 accelerator: DWAFINAL
 last load: 2011/07/29 07:01:04

AQT name	FactTab	#tab	#matched	address
aqt56d5aea7-32f4-44e6-8d98-02a7af37630f	103	1	845	0x4d84ac70
aqt03ec4c20-7ba8-4c3a-ae56-4134b005269d	103	2	27	0x4d95c298
aqt4ae7c2fd-5b94-423d-bc49-9ca3f5f38799	103	2	3912	0x4d84adc8
aqt5ed69a75-15e3-45cc-9892-4f5386257895	103	3	83	0x4d95c4a0
aqtdf314aa6-177d-4443-9f6d-f14ba766995a	103	3	37	0x4d95c028
aqt7e36b1f2-4646-4075-ac0b-5fdee475cd7e	103	4	518	0x4d95c758

mart: school3
 accelerator: DWAFINAL
 last load: 2011/07/29 07:01:50

AQT name	FactTab	#tab	#matched	address
aqt92b36a8a-1567-4146-833c-385cd103f5d4	103	1	678	0x4d95cac0
aqt3189bec1-b6c9-417d-b969-92c687ef2e44	103	2	59	0x4d95cc18
aqt8d3b3dc8-59b6-4e34-822b-75b06b99c900	103	2	4487	0x4d90c0d8
aqt5f9c2a05-9131-4738-a929-036fcf77f65c	103	3	71	0x4d90c2e0
aqtee08ed16-6a5c-4478-ac57-fc4f99539c74	103	3	795	0x4d95ce20
aqt04d1c96a-022b-4ed7-938d-caf765bc9926	103	4	367	0x4d90c598

18 entries

Figure 20-15. **onstat -g aqt** command output

If you use the AQT name for the optional *aqt_name* parameter, the command prints information about the specific AQT.


```

AQT: aqt6de1afdd-f10a-45b0-93e9-0c208405fefd
database: iwadb
AQT tabid: 125
Fact table: 111
Number of times matched: 8947

Join structure: alias(tabid)[colno,...] = alias(tabid)[colno,...] {u:unique}
0(111)[1] = 1(110)[1] u
           1(110)[2] = 2(109)[1] u
                     2(109)[5] = 3(101)[1] u
                               3(101)[3] = 4(100)[1] u

0(111)[2] = 5(106)[1] u
           5(106)[2] = 6(103)[1] u
           5(106)[3] = 7(104)[1] u
           5(106)[4] = 8(105)[1] u
                     8(105)[3] = 9(101)[1] u
                               9(101)[3] = 10(100)[1] u

           5(106)[5] = 11(102)[1] u
0(111)[2,3] = 15(108)[1,2] u
              15(108)[1] = 16(106)[1] u
                        16(106)[2] = 17(103)[1] u
                        16(106)[3] = 18(104)[1] u
                        16(106)[4] = 19(105)[1] u
                                19(105)[3] = 20(101)[1] u
                                        20(101)[3] = 21(100)[1] u

                        16(106)[5] = 22(102)[1] u
15(108)[2] = 23(107)[1] u
            23(107)[2] = 24(101)[1] u
                    24(101)[3] = 25(100)[1] u

0(111)[3] = 12(107)[1] u
           12(107)[2] = 13(101)[1] u
                     13(101)[3] = 14(100)[1] u

```

Figure 20-16. **onstat -g aqt aqt_name** command output

Output description

The AQTs are grouped by the data mart that they belong to. The groups are sorted by accelerator name, and then by data mart name. Within the data mart groups, the AQTs are sorted in the following order: Fact table tabid (FactTab), number of tables (#tab), and AQT name.

The output comes from the entries in the dictionary cache that refer to the AQTs of the data marts. The output is shown only if the AQTs have been loaded into the dictionary cache, which normally occurs when a query is being matched against the AQTs.

Before the server attempts to match a query against the AQTs, the AQTs do not have any entries in the dictionary cache. The **onstat -g aqt** command will not show any entries in the output. When the dictionary cache is initialized during the database server startup, the columns #matched and address get new values.

The **onstat -g aqt** command prints the following information:

mart The name of the data mart

accelerator

 The name of the accelerator instance

last load

 The time stamp for when the data mart was last loaded

AQT name

The unique system-generated name of the AQT

FactTab

The tabid of the fact table for the AQT

#tab The number of tables that are part of the AQT

#matched

The counter for query matches that have occurred for the AQT

address

The internal database server memory address for the AQT

The **onstat -g aqt aqt_name** command prints the following information:

AQT The unique system-generated name of the AQT

database

The name of the database to which the AQT belongs

AQT tabid

The tabid for the entry that constitutes the AQT in the systables system catalog table of the database server.

Fact table

The tabid of the fact table of the AQT

Number of times matched

The counter for query matches that have occurred for the AQT

The information about the AQT is followed by a textual representation of the star schema of the data mart. The textual representation shows how the columns of the tables are related to each other in the star join.

For information about the Informix Warehouse Accelerator, see the *IBM Informix Warehouse Accelerator Administration Guide*.

onstat -g arc command: Print archive status

Use the **onstat -g arc** command to display information about the last committed archive for each dbspace and also information about any current ongoing archives.

Syntax:

►►onstat— -g—arc—————►►

Example output

Dbspaces - Ongoing archives								
number	name	Q Size	Q Len	buffer	partnum	size	Current-page	
1	rootdb	100	3	100	0x1001c9	0	1:128	
3	datadbs01	0	0					
4	datadbs02	0	0					

Dbspaces - Archive Status						
name	number	level	date	log	log-position	
rootdb	1	0	07/30/2009.09:59	28	0x320018	
datadbs01	3	0	07/30/2009.09:59	28	0x320018	
datadbs02	4	0	07/30/2009.09:59	28	0x320018	

Figure 20-17. **onstat -g arc** command output

Output description - Ongoing archives

This output section represents current information about the archives. If no archives are active in the system, this section is not displayed.

Column	Description
Number	The number of the dbspace
Name	The name of the dbspace
Q Size	The before-image queue list size. This information is primarily for IBM support.
Q Len	The before-image queue length. This information is primarily for IBM support.
Buffer	The number of pages used in the before-image buffer
Partnum	The partition number of the before-image bin
Size	The number of pages in the before-image bin
Current-page	The current page that is being archived

Note: The before-image bin is a temporary table created in a temporary dbspace, or in the root dbspace if you do not have any temporary dbspaces. If the before-image bin becomes too small, it can extend to additional partitions, in which case the output will display see multiple Partnum and Size fields for the same dbspace.

Output description - Archive status

This output section contains information about the last backup that has occurred for each dbspace.

Column	Description
Name	The name of the dbspace
Number	The dbspace number
Level	The archive level
Date	The date and time of the last archive

Column	Description
Log	The unique ID (UNIQID) of the checkpoint that was used to start the archive
Log-position	The log position (LOGPOS) of the checkpoint that was used to start the archive

onstat -g ath command: Print information about all threads

Use the **onstat -g ath** command to display information about all threads.

Syntax:

►► onstat — -g — ath ————— ►►

Example output

Threads:						
tid	tcb	rstcb	prty	status	vp-class	name
2	10bbf36a8	0	1	IO Idle	3lio	lio vp 0
3	10bc12218	0	1	IO Idle	4pio	pio vp 0
4	10bc31218	0	1	running	5aio	aio vp 0
5	10bc50218	0	1	IO Idle	6msc	msc vp 0
6	10bc7f218	0	1	running	7aio	aio vp 1
7	10bc9e540	10b231028	1	sleeping secs: 1	1cpu	main_loop()
8	10bc12548	0	1	running	1cpu	tlitcpoll
9	10bc317f0	0	1	sleeping forever	1cpu	tlitcplst
10	10bc50438	10b231780	1	IO Wait	1cpu	flush_sub(0)
11	10bc7f740	0	1	IO Idle	8aio	aio vp 2
12	10bc7fa00	0	1	IO Idle	9aio	aio vp 3
13	10bd56218	0	1	IO Idle	10aio	aio vp 4
14	10bd75218	0	1	IO Idle	11aio	aio vp 5
15	10bd94548	10b231ed8	1	sleeping forever	1cpu	aslogflush
16	10bc7fd00	10b232630	1	sleeping secs: 34	1cpu	btscanner 0
32	10c738ad8	10b233c38	1	sleeping secs: 1	1cpu	onmode_mon
50	10c0db710	10b232d88	1	IO Wait	1cpu	sqlxec

Figure 20-18. onstat -g ath command output

Output description

tid	Thread ID
tcb	Thread control block access
rstcb	RSAM thread control block access
prty	Thread priority
status	Thread status
vp-class	Virtual processor class
name	Thread name

Related reference:

“**onstat -g wst** command: Print wait statistics for threads” on page 20-185

“NUMFDSERVERS configuration parameter” on page 1-102

onstat -g buf command: Print buffer pool profile information

Use the **onstat -g buf** command to display profile information for each buffer pool.

Syntax:

►► onstat — -g buf ————— ►►

Example output

Profile							
Buffer pool page size: 2048							
dskreads	pagreads	bufreads	%cached	dskwrits	pagwrits	bufwrits	%cached
53056	118251	982617385	99.99	10640768	10915965	15968877	33.37
bufwrits_sinceckpt		bufwaits	ovbuff	flushes			
17477		863	0	874			
Fg Writes		LRU Writes	Avg. LRU Time		Chunk Writes		
0		0	NaN		110767		
Fast Cache Stats							
gets	hits	%hits	puts				
48314472	47220455	97.74	45757549				

Figure 20-19. **onstat -g buf** command output

Output description

Buffer pool page size

Number of bytes in a page in the buffer pool being profiled

dskreads

Number of disk read operations performed to bring pages into this buffer pool. Each read operation reads one or more pages.

pagreads

Number of pages read from disk to this buffer pool

bufreads

Number of times a memory image for a page was read from this buffer pool

%cached

Percentage of page reads for this buffer pool that were satisfied by a cached page image (rather than having to perform a disk read). Computed as $(\text{bufreads} - \text{dskreads}) / \text{bufreads} \times 100$. Higher percentages indicate better caching performance.

dskwrits

Number of disk write operations performed to write changed pages from this buffer pool back to disk. Each write operation writes one or more pages.

pagwrits

Number of pages written to disk from this buffer pool

bufwrits

Number of times a memory image of a page was written to in this buffer pool

%cached

Percentage of page writes for this buffer pool that were satisfied by a cached page image (rather than having to perform a disk write). Computed as $(\text{bufwrits} - \text{dskwrits}) / \text{bufwrits} \times 100$.

bufwrits_sinceckpt

Number of times a memory image of a page was written to in this buffer pool since the last checkpoint

bufwaits

Number of times a thread had to wait for a lock on a buffer in this buffer pool. Higher numbers indicate more contention among multiple threads for mutually incompatible locks on the same pages.

ovbuff

Number of times a changed buffer from this buffer pool was written to disk specifically to create a free buffer to read another requested page. If the *ovbuff* value is high, it may indicate that the buffer pool is not large enough to hold the working set of pages needed by the applications using this buffer pool, which may lead to performance degradation from I/O thrashing.

flushes

Number of times the server performed a mass flush of all dirty buffers in the buffer pool. This can occur for a variety of reasons, including as part of checkpoint processing or if the buffer pool is running out of clean buffers despite normal LRU cleaning activity.

Fg Writes

Number of changed buffers from this buffer pool that were written to disk by a non-I/O flusher thread that was accessing the buffer. This number is a superset of *ovbuff*. In addition to the writes to service page faults counted by *ovbuff*, this value also includes foreground writes that are done by certain operations to maintain the consistency of database logs and reserved pages in order to guarantee correctness of crash recovery in special cases.

LRU Writes

Number of changed buffers from this buffer pool that were written to disk by an LRU cleaner thread. LRU cleaners are activated if the buffer pool exceeds the value specified in the **lru_max_dirty** field of the BUFFERPOOL configuration parameter or if foreground writes occur due to buffer pool overflows.

Avg. LRU Time

Average amount of time taken by an LRU cleaner thread to clean a single LRU chain.

Chunk Writes

Number of changed buffers that were written to disk by a chunk cleaning operation. Chunk cleaning writes out all changed buffers of a given chunk that are in the buffer pool. This is done in a variety of special circumstances that need to clean a large number of buffers quickly, such as checkpoint processing and fast recovery.

Fast Cache Stats

Statistics for the fast cache, which is a type of cache that reduces the time needed for accessing the bufferpool

- gets** The number of times the server looked for a buffer in the fast cache
- hits** The number of times that the server found the buffer it was searching for in the fast cache
- %hits** The percentage of hits, which is $\text{hits} \times 100 / \text{gets}$
- puts** The number of times that the server inserted buffers inserted into the fast cache

onstat -g cat command: Print ER global catalog information

Use the **onstat -g cat** command to display information from the Enterprise Replication global catalog. The global catalog contains information about the defined servers, replicates, and replicate sets on each of the servers within the domain.

Syntax:

►►—onstat— -g—cat—►►

If a replicated table is undergoing an alter operation, the **onstat -g cat** command shows that it is in alter mode. For example, use this command to determine:

- How many servers and how many replicates are configured
- Which table matches a given replicate
- Whether a server is a root or leaf server
- The current bitmap mask for a given server. You can use the bitmap mask with the output from the **onstat -g rqm** command to determine which server Enterprise Replication is waiting on for an acknowledgment.

The **onstat -g cat** command has the following formats:

```
onstat -g cat
onstat -g cat replname
onstat -g cat scope
```

The following table describes these formats.

Modifier	Description
<i>replname</i>	Specify the name of a replicate
<i>scope</i>	Specify one of the following values: <ul style="list-style-type: none">servers : Displays information only about serversrepls : Displays information only about replicatesfull : Displays expanded information for both replicate servers and replicates

Example output

This sample output from the **onstat -g cat repls** command shows that the table **tab** is in alter mode. The replicate **rep1** is defined on this table, its replicate ID is 6553601. For more information on the replicate attributes that this command

displays, see the *IBM Informix Enterprise Replication Guide*.

```
GLOBAL-CATALOG CACHE STATISTICS
REPLICATES
-----
Parsed statements:
  Id 6553601 table tab
  Id 6553602 table tab12
Inuse databases: test(2)
  Name: rep1, Id: 6553601 State: ACTIVE Flags: 0x800000 ALTERMODE
    use 0 lastexec Wed Dec 31 18:00:00 1969
    Local Participant: test:nagaraju.tab
    Attributes: TXN scope, Enable ATS, Enable RIS, all columns
      sent in updates
    Conflict resolution: [TIMESTAMP]
    Column Mapping: ON, columns INORDER, offset 8, uncomp_len 12
    Column Name Verification: ON
    No Replicated UDT Columns
  Name: rep12, Id: 6553602 State: ACTIVE Flags: 0x800000 use 0
    lastexec Wed Dec 31 18:00:00 1969
    Local Participant: test:nagaraju.tab12
    Attributes: TXN scope, Enable ATS, Enable RIS, all columns
      sent in updates
    Conflict resolution: [TIMESTAMP]
    Column Mapping: ON, columns INORDER, offset 8, uncomp_len 2064
    Column Name Verification: ON
    No Replicated UDT Columns
```

Figure 20-20. **onstat -g cat repls** command output

onstat -g cdr command: Print ER statistics

Use the **onstat -g cdr** command to display the output for all of the Enterprise Replication statistics commands.

Syntax:

►► onstat — - g — cdr —————►►

The output of the **onstat -g cdr** command is a combination of the following Enterprise Replication **onstat** command outputs:

- **onstat -g cat**
- **onstat -g grp**
- **onstat -g que**
- **onstat -g rqm**
- **onstat -g nif all**
- **onstat -g rcv**
- **onstat -g dss**
- **onstat -g dtc**
- **onstat -g rep**

For examples and descriptions of the output, refer to the following topics about the commands that compose the **onstat -g cdr** output:

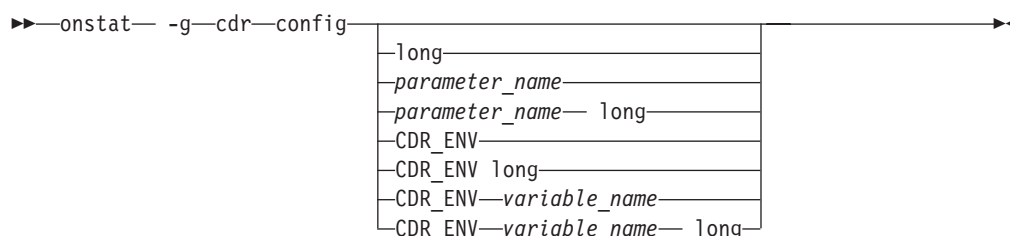
- “**onstat -g cat** command: Print ER global catalog information” on page 20-57
- “**onstat -g grp** command: Print ER grouper statistics” on page 20-91
- “**onstat -g que** command: Prints ER queue statistics” on page 20-140

- “**onstat -g rqm** command: Print low-level queue statistics” on page 20-148
- “**onstat -g nif** command: Print statistics about the network interface” on page 20-120
- “**onstat -g rcv** command: Print ER receive manager statistics” on page 20-144
- “**onstat -g dss** command: Print ER environment data” on page 20-84
- “**onstat -g dtc** command: Print delete table cleaner statistics” on page 20-85
- “**onstat -g rep** command: Print ER schedule manager events” on page 20-148

onstat -g cdr config command: Print ER settings

Use the **onstat -g cdr config** command to display the settings of Enterprise Replication configuration parameters and environment variables that can be set with the CDR_ENV configuration parameter.

Syntax:



This command has the following formats:

```
onstat -g cdr config
onstat -g cdr config long
onstat -g cdr config parameter_name
onstat -g cdr config parameter_name long
onstat -g cdr config CDR_ENV
onstat -g cdr config CDR_ENV long
onstat -g cdr config CDR_ENV variable_name
onstat -g cdr config CDR_ENV variable_name long
```

The **long** option prints additional information about settings that can be useful for IBM Support.

The following table describes *parameter_name* and *variable_name*.

Modifier	Description
<i>parameter_name</i>	The name of an Enterprise Replication configuration parameter
<i>variable_name</i>	The name of an Enterprise Replication environment variable

If you use **onstat -g cdr config** without any options, the settings of all Enterprise Replication configuration parameters and environment variables are included in the output. If you specify the CDR_ENV configuration parameter without an environment variable name, all Enterprise Replication environment variables are included in the output.

Example output

The following sample output of the **onstat -g cdr config ENCRYPT_CDR** command shows the setting of the ENCRYPT_CDR configuration parameter:

```
onstat -g cdr config ENCRYPT_CDR

ENCRYPT_CDR configuration setting:          0
```

The following sample output of the **onstat -g cdr config CDR_ENV** command shows the settings of all Enterprise Replication environment variables:

```
onstat -g cdr config CDR_ENV

CDR_ENV environment variable settings:
  CDR_LOGDELTA:
    CDR_LOGDELTA configuration setting:      0
  CDR_PERFLOG:
    CDR_PERFLOG configuration setting:       0
  CDR_ROUTER:
    CDR_ROUTER configuration setting:        0
  CDR_RMSCALEFACT:
    CDR_RMSCALEFACT configuration setting:   0
  CDRSITES_731:
    CDRSITES_731 configuration setting:      [None configured]
  CDRSITES_92X:
    CDRSITES_92X configuration setting:      [None configured]
  CDRSITES_10X:
    CDRSITES_10X configuration setting:      [None configured]
```

The following sample output of the **onstat -g cdr config** command shows the settings of all Enterprise Replication configuration parameters and CDR_ENV environment variables:

```
onstat -g cdr config

CDR_DBSPACE:
  CDR_DBSPACE configuration setting:         rootdbs
CDR_DSLOCKWAIT:
  CDR_DSLOCKWAIT configuration setting:      5
CDR_EVALTHREADS:
  CDR_EVALTHREADS configuration setting:     1, 2
CDR_MAX_DYNAMIC_LOGS:
  CDR_MAX_DYNAMIC_LOGS configuration setting: 0
CDR_NIFCOMPRESS:
  CDR_NIFCOMPRESS configuration setting:     0
CDR_QDATA_SBSpace:
  CDR_QDATA_SBSpace configuration setting:   cdrsbsp
CDR_QHDR_DBSPACE:
  CDR_QHDR_DBSPACE configuration setting:    rootdbs
CDR_QUEUEMEM:
  CDR_QUEUEMEM configuration setting:        4096
CDR_SERIAL:
  CDR_SERIAL configuration setting:          0, 0
CDR_SUPPRESS_ATSRISWARN:
  CDR_SUPPRESS_ATSRISWARN configuration setting: [None suppressed]
ENCRYPT_CDR:
  ENCRYPT_CDR configuration setting:          0
ENCRYPT_CIPHERS:
  ENCRYPT_CIPHERS configuration setting:      [None configured]
ENCRYPT_MAC:
  ENCRYPT_MAC configuration setting:          [None configured]
ENCRYPT_MACFILE:
  ENCRYPT_MACFILE configuration setting:      [None configured]
ENCRYPT_SWITCH:
  ENCRYPT_SWITCH configuration setting:       0,0
CDR_ENV environment variable settings:
  CDR_LOGDELTA:
    CDR_LOGDELTA configuration setting:      0
  CDR_PERFLOG:
    CDR_PERFLOG configuration setting:       0
  CDR_ROUTER:
```

```

CDR_ROUTER configuration setting:          0
CDR_RMSCALEFACT:
CDR_RMSCALEFACT configuration setting:     0
CDRSITES_731:
CDRSITES_731 configuration setting:        [None configured]
CDRSITES_92X:
CDRSITES_92X configuration setting:        [None configured]
CDRSITES_10X:
CDRSITES_10X configuration setting:        [None configured]

```

onstat -g ckp command: Print checkpoint history and configuration recommendations

Use the **onstat -g ckp** command to print checkpoint history and display configuration recommendations if a suboptimal configuration is detected.

Syntax:

```

▶▶ onstat -g ckp ◀◀

```

Example output

Auto Checkpoints=On RTO_SERVER_RESTART=60 seconds Estimated recovery time 7 seconds											
				Critical Sections							
Interval	Clock	Trigger	LSN	Total Time	Flush Time	Block Time	# Waits	Ckpt Time	Wait Time	Long Time	#Dirty Buffers
1	18:41:36	Startup	1:f8	0.0	0.0	0.0	0	0.0	0.0	0.0	4
2	18:41:49	Admin	1:11c12cc	0.3	0.2	0.0	1	0.0	0.0	0.0	2884
3	18:42:21	Llog	8:188	2.3	2.0	2.0	1	0.0	2.0	2.0	14438
4	18:42:44	User	10:19c018	0.0	0.0	0.0	1	0.0	0.0	0.0	39
5	18:46:21	RTO	13:188	54.8	54.2	0.0	30	0.6	0.4	0.6	68232
				Physical Log Logical Log							
Dskflu	Total	Avg	Total	Avg							
/Sec	Pages	/Sec	Pages	/Sec							
4	3	0	1	0							
2884	1966	163	4549	379							
7388	318	10	65442	2181							
39	536	21	20412	816							
1259	210757	1033	150118	735							
Max Plog	Max Llog	Max Dskflush	Avg Dskflush		Avg Dirty	Blocked					
pages/sec	pages/sec	Time	pages/sec		pages/sec	Time					
8796	6581	54	43975		2314	0					

Figure 20-21. onstat -g ckp command output

Output description

Auto Checkpoints

Indicates if the AUTO_CKPTS configuration parameter is on or off

RTO_SERVER_RESTART

Displays the RTO time in seconds. Zero (0) means that RTO is off.

Estimated recovery time ## seconds

Indicates the estimated recovery time if the data server stops responding. This value only appears if RTO_SERVER_RESTART is active.

Interval

Checkpoint interval ID

Clock Time

Clock time when checkpoint occurred.

Trigger

Event that triggered the checkpoint. An asterisk (*) indicates that the checkpoint requested was a transaction-blocking checkpoint.

Trigger name	Description
Admin	Administrator related tasks. For example: <ul style="list-style-type: none">• Create, drop, or rename a dbspace• Add or drop a chunk• Add or drop a log file• Change physical log size or location• After "shrink" operation on partition• Turn on or off mirroring
Backup	Backup related operations. For example: <ul style="list-style-type: none">• Fake backup• Start of an archive• After the completion of a physical restore
CDR	ER subsystem is started for the first time, or is restarted after all of the replication participants were removed.
CKPTINTVL	When the checkpoint interval expires. The checkpoint interval is the value specified for the CKPTINTVL parameter in the ONCONFIG file.
Conv/Rev	Conversion reversion checkpoint. After the check phase of convert and before the actual conversion of disk structures. After the reversion is completed also triggers a checkpoint.
HA	High availability. For example: <ul style="list-style-type: none">• A new RSS or SDS node is added to a High Availability cluster• A secondary server is promoted to a primary server• The physical log file is low on a secondary server
HDR	High-Availability Data Replication. For example: <ul style="list-style-type: none">• The mode of the server is changed• The start of the first transfer after HDR is setup• There is the potential for a physical log overflow on primary or secondary servers
Lightscan	Before the look aside is turned off on partitions.
Llog	Running out of logical log resources.

Trigger name	Description
LongTX	Long Transaction. If a long transaction has been found but not stopped, a checkpoint is initiated to stop the transaction. During rollback, a checkpoint is initiated in the rollback phase if a checkpoint has not already happened after long transaction has been aborted.
Misc	Miscellaneous events. For example: <ul style="list-style-type: none"> • A dbspace or chunk is being brought down because of I/O errors • During rollback when the addition of the chunk is being undone. For example, when removing the chunk.
Pload	When the High-Performance Loader starts in the Express mode.
Plog	Physical log has one of the following conditions: <ul style="list-style-type: none"> • Physical log is 75% full • The amount of physical log used plus the number of dirty partitions is more than 90% of physical log size
Restore Pt	Restore Point. Checkpoints at the start and end of a restore point. The restore point is (used by conversion guard) CONVERSION_GUARD configuration parameter is enabled and a temporary directory is specified in the RESTORE_POINT_DIR configuration parameter.
Recovery	During a restore, at the start of a fast recovery.
Reorg	At the start of online index build.
RTO	Maintaining the Recovery Time Objective (RTO) policy. During normal operations, when the restart time after a crash might exceed the value set for the RTO_SERVER_RESTART configuration parameter.
Stamp Wrap	Checkpoint timestamp. If the new checkpoint timestamp appears to be before the last written checkpoint, then the timestamp is advanced out of interval between checkpoints. Another checkpoint is triggered.
Startup	At the startup of the database server.
Uncompress	Uncompress commands that are issued on a table or partition. This applies only for checkpoints on tables or databases that are not logged.
User	A checkpoint request is submitted by the user.

LSN Logical log position where checkpoint is recorded

Total Time

Total checkpoint duration, in seconds, from request time to checkpoint completion

Flush Time

Time, in seconds, to flush bufferpools

Block Time

Time a transaction was blocked, in seconds, by a checkpoint that was triggered by a scarcity of some needed resource. For example, running out of physical log, or wrap-around of the logical log.

Waits

Number of transactions blocked waiting for checkpoint

Ckpt Time

Time, in seconds, for all transactions to recognize a requested checkpoint

Wait Time

Average time, in seconds, that transactions waited for checkpoint

Long Time

Longest amount of time, in seconds, a transaction waited for checkpoint

Dirty Buffers

Number of dirty buffers flushed to disk during checkpoint

Dskflu/sec

Number of buffers flushed per second

Physical Log Total Pages

Total number of pages physically logged during checkpoint interval

Physical Log Avg/Sec

Average rate of physical log activity during checkpoint interval

Logical Log Total Pages

Total number of pages logically logged during checkpoint interval

Logical Log Avg/Sec

Average rate of logical log activity during checkpoint interval

Max Plog pages/sec

Maximum rate of physical log activity during checkpoint interval

Max Llog pages/sec

Maximum rate of logical log activity during checkpoint interval

Max Dskflush Time

Maximum time, in seconds, to flush bufferpools to disk

Avg Dskflush pages/sec

Average rate bufferpools are flushed to disk

Avg Dirty pages/sec

Average rate of dirty pages between checkpoints

Blocked Time

Longest blocked time, in seconds, since the database server was last started

Performance advisory messages

If the Informix data server detects a configuration that is less than optimal, a performance advisory message with tuning recommendations appears below the

checkpoint history. This performance advisory message also appears in the message log. Following are examples of performance advisory messages:

Physical log is too small for bufferpool size. System performance may be less than optimal.

Increase physical log size to at least %ldKb

Physical log is too small for optimal performance.

Increase the physical log size to at least \$ldKb.

Logical log space is too small for optimal performance.

Increase the total size of the logical log space to at least %ld Kb.

Transaction blocking has taken place. The physical log is too small.

Please increase the size of the physical log to %ldKb

Transaction blocking has taken place. The logical log space is too small.

Please increase the size of the logical log space to %ldKb

onstat -g cluster command: Print high-availability cluster information

Use the **onstat -g cluster** command to display information about the servers in a high-availability cluster environment.

Syntax:

```
▶▶ onstat -g cluster [verbose] ▶▶
```

The **onstat -g cluster** command combines the functionality of **onstat -g dri**, **onstat -g sds**, and **onstat -g rss**. The output of the **onstat -g cluster** command differs slightly depending on whether the command is run on the primary server or on one of the secondary servers.

Example output (primary server)

Following is sample output from the **onstat -g cluster** command. The sample shows output when the command is run on the primary server.

```
Primary Server:serv1
Current Log Page:16,476
Index page logging status: Enabled
Index page logging was enabled at: 2010/01/11 14:05:17

Server ACKed Log      Supports      Status
      (log, page)    Updates
serv2 16,476         Yes          SYNC(SDS),Connected,Active
serv3 16,476         Yes          ASYNC(HDR),Connected,On
serv4 16,476         Yes          ASYNC(RSS),Connected,Active
```

Figure 20-22. **onstat -g cluster** command output (run on the primary server)

Output description (primary server)

Primary server

The name assigned to the primary server.

Current log page

The log ID and page number of the current log page.

Index page logging status

Indicates whether index page logging is enabled or disabled.

Index page logging was enabled at

The date and time that index page logging was enabled.

Server The name of the secondary server.

ACKed Log (log, page)

The log ID and page number of the last acknowledged log transmission.

Supports Updates

Displays whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the UPDATABLE_SECONDARY configuration parameter).

Status Displays the connection status of the secondary server

Example output (primary server, verbose output)

Following is sample output from the **onstat -g cluster verbose** command. The sample shows output when the command is run on the primary server with the verbose option.

```
Primary Server:serv1
Current Log Page:16,479
Index page logging status: Enabled
Index page logging was enabled at: 2010/01/11 14:05:17
```

```
-----
server name: serv3
type: ASYNC (HDR)
control block: 0x4b673018
server status: On
connection status: Connected
Last log page sent (log id, page): 16,479
Last log page acked (log id, page); 16,479
Approximate log page backlog: 0
SDS cycle not used
Delayed Apply Not Used
Stop Apply Not Used
Time of last ack: 2010/01/15 14:09:12
Supports Updates: Yes
```

```
-----
server name: serv2
type: SYNC (SDS)
control block: 0x4c2de0b8
server status: Active
connection status: Connected
Last log page sent (log id, page): 16,479
Last log page acked (log id, page); 16,479
Approximate log page backlog: 0
SDS cycle current: 20 ACKed: 20
Delayed Apply Not Used
Stop Apply Not Used
Time of last ack: 2010/01/15 14:09:13
Supports Updates: Yes
```

Figure 20-23. **onstat -g cluster verbose** command output (run on the primary server)

Output description (primary server, verbose output)

Primary server

The name of the primary server

Current log page

The log ID and page number of the current log page.

Index page logging status

Indicates whether index page logging is enabled or disabled.

Index page logging was enabled at

The date and time that index page logging was enabled.

Server name

The name of the secondary server.

type

Displays whether the secondary server is connected synchronously (SYNC) or asynchronously (ASYNC). Also displays the type of secondary server: HDR, SDS, or RSS.

control block

The in-memory address of the thread control block.

server status

Displays the current status of the secondary server.

connection status

Displays the current network connection status of the secondary server.

Last log page sent (log id, page)

The log ID and page number of the most recent log page sent by the primary server to the secondary server.

Last log page acked (log id, page)

The log ID and page number of the most recent log page the secondary server acknowledged.

Approximate log page backlog

Indicates the approximate number of log pages that have yet to be processed by the secondary server.

SDS cycle

Indicates the cycle number to which the primary server has advanced and which the shared disk secondary server has acknowledged. Used internally by IBM support to monitor coordination of the primary server with the secondary server.

Delayed Apply

Indicates whether the secondary server waits for a specified amount of time before applying logs (as specified by the DELAY_APPLY configuration parameter).

Stop Apply

Indicates whether the secondary server has stopped applying log files received from the primary server (as specified by the STOP_APPLY configuration parameter).

Time of last ack

The date and time of the last acknowledged log.

Supports Updates

Displays whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the UPDATABLE_SECONDARY configuration parameter).

Example output (secondary server)

Following is sample output from the **onstat -g cluster** command. The sample shows output when the command is run on a secondary server.

```
Primary Server:serv1
Index page logging status: Enabled
Index page logging was enabled at: 2010/01/11 14:05:17

Server ACKed Log      Supports      Status
      (log, page) Updates
serv2 16,479      Yes      SYNC(SDS),Connected,Active
```

Figure 20-24. **onstat -g cluster** command output (run on the secondary server)

Output description (secondary server)

Primary server

The name of the primary server

Index page logging status

Indicates whether index page logging is enabled or disabled.

Index page logging was enabled at

The date and time that index page logging was enabled.

Server The name of the secondary server.

ACKed Log (log, page)

The log ID and page number of the last acknowledged log.

Supports Updates

Displays whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the `UPDATABLE_SECONDARY` configuration parameter).

Status Displays the connection status of the secondary server.

Related reference:

“`DELAY_APPLY` Configuration Parameter” on page 1-53

“`STOP_APPLY` Configuration Parameter” on page 1-145

“`UPDATABLE_SECONDARY` Configuration Parameter” on page 1-152

onstat -g cmsm command: Print Connection Manager information

Use the **onstat -g cmsm** command to display Connection Manager information, such as the number of Connection Manager daemons attached to the current database instance, as well as the number of connections each daemon has processed.

Syntax:

```
▶▶ onstat -g cmsm [cm_name] ▶▶
```

The optional *cm_name* parameter is used to specify the name of a Connection Manager instance. Omitting the *cm_name* parameter displays information about all active Connection Manager instances.

Example output

```
Connection Manager Name: cm1      Hostname: apollo11
Cluster c1
  SLA      Connections  Service/Protocol  Rule
  oltp1    3           9696/onsoctcp    DBSERVERS=primary

Failover Configuration:
Connection Manager name      Rule      Timeout  State
cm1                          SDS+HDR+RSS      0  Active Arbitrator, Primary is up

Grig grid1
  SLA      Connections  Service/Protocol  Rule
  gridany  3           9400/onsoctcp    DBSERVERS=ANY
  grid1_avail 30       9410/onsoctcp    DBSERVERS=(ny1,ny2,ny3),\
                                           (ca1,ca2,ca3)
```

Figure 20-25. **onstat -g cmsm** command output with the Connection Manager managing one cluster and one grid

Output description

The output of the **onstat -g cmsm** command is divided into several sections. The first section displays the Connection Manager instance name and host name. The rest of the sections display service level agreements and failover configurations for each connection unit. The example output shows the Connection Manager configured with two connection units named **c1** and **grid1**.

Connection Manager Name

The name of the Connection Manager instance.

Hostname

The name of the host.

SLA The name of the service level agreement.

Connections

The number of connections that are processed with the service level agreement.

Service/Protocol

The port number or service name associated with the Service Level Agreement, followed by the protocol (onsoctcp, drsoctcp, onsocssl, drsocssl).

Rule The service level agreement definition which can include server types (Primary, SDS, RSS, HDR) or server names.

Failover Configuration

Connection Manager Name

The name of the Connection Manager instance.

Rule The failover configuration.

Timeout

The timeout value, in seconds, for the failover configuration.

State An indicator of the status of the primary server. Also indicates the Connection Manager instance serving as the failover arbitrator.

onstat -g con command: Print condition and thread information

Use the **onstat -g con** command to display information about conditions and the threads that are waiting for the conditions.

Syntax:

►► onstat — -g — con ————— ◀◀

Example output

Conditions with waiters:				
cid	addr	name	waiter	waittime
271	c63d930	netnorm	1511	6550

Figure 20-26. **onstat -g con** command output

Output description

cid Condition identifier

addr Condition control block address

name Name of condition the thread is waiting on

waiter ID of thread waiting on condition

waittime

Time, in seconds, thread has been waiting on this condition

onstat -g cpu: Print runtime statistics

Use the **onstat -g cpu** command to display information about runtime statistics for each thread that is running in the server.

Syntax:

►► onstat — -g — cpu ————— ◀◀

Example output

```
onstat -g cpu
```

Thread CPU Info:

tid	name	vp	Last Run	CPU Time	#scheds	status
2	lio vp 0	3lio*	07/18 08:35:35	0.0000	1	IO Idle
3	pio vp 0	4pio*	07/18 08:35:36	0.0102	2	IO Idle
4	aio vp 0	5aio*	07/18 08:35:47	0.6876	68	IO Idle
5	msc vp 0	6msc*	07/18 11:47:24	0.0935	14	IO Idle
6	main_loop()	1cpu*	07/18 15:02:43	2.9365	23350	sleeping secs: 1
7	soctcpoll	7soc*	07/18 08:35:40	0.1150	1	running
8	soctcpio	8soc*	07/18 08:35:40	0.0037	1	running
9	soctcplst	1cpu*	07/18 11:47:24	0.1106	10	sleeping forever
10	soctcplst	1cpu*	07/18 08:35:40	0.0103	6	sleeping forever
11	flush_sub(0)	1cpu*	07/18 15:02:43	0.0403	23252	sleeping secs: 1
12	flush_sub(1)	1cpu*	07/18 15:02:43	0.0423	23169	sleeping secs: 1
13	flush_sub(2)	1cpu*	07/18 15:02:43	0.0470	23169	sleeping secs: 1
14	flush_sub(3)	1cpu*	07/18 15:02:43	0.0407	23169	sleeping secs: 1
15	flush_sub(4)	1cpu*	07/18 15:02:43	0.0307	23169	sleeping secs: 1
16	flush_sub(5)	1cpu*	07/18 15:02:43	0.0323	23169	sleeping secs: 1
17	flush_sub(6)	1cpu*	07/18 15:02:43	0.0299	23169	sleeping secs: 1
18	flush_sub(7)	1cpu*	07/18 15:02:43	0.0314	23169	sleeping secs: 1
19	kaio	1cpu*	07/18 14:56:42	1.4560	2375587	IO Idle
20	aslogflush	1cpu*	07/18 15:02:43	0.0657	23166	sleeping secs: 1
21	btscanner_0	1cpu*	07/18 15:00:53	0.0484	784	sleeping secs: 61
37	onmode_mon	1cpu*	07/18 15:02:43	0.3467	23165	sleeping secs: 1
43	dbScheduler	1cpu*	07/18 14:58:14	1.6613	320	sleeping secs: 31
44	dbWorker1	1cpu*	07/18 13:48:10	0.4264	399	sleeping forever
45	dbWorker2	1cpu*	07/18 14:48:11	1.9346	2936	sleeping forever
94	bf_priosweep()	1cpu*	07/18 15:01:42	0.0431	77	cond wait bp_cond

Figure 20-27. `onstat -g cpu` command output

Output description

tid The ID of the thread

name The name of the thread

vp The ID of the virtual processor in which the thread is running

Last Run

The timestamp when the thread last ran

CPU Time

The time taken until now by the thread

#scheds

The number of times the thread was scheduled to run

status The status of the thread. Possible status values are:

- cond wait
- IO Idle
- join wait
- mutex wait
- ready
- sleeping
- terminated
- running
- yield

onstat -g dbc command: Print dbScheduler and dbWorker thread statistics

Use the **onstat -g dbc** command to display statistics about the Scheduler tasks that are currently running, which are handled by dbWorker threads, or scheduled to be run, which are handled by the dbScheduler thread.

Syntax:

►► onstat — -g — dbc —————►►

Example output

```
Worker Thread(0) 46fa6f10
=====
Task: 47430c18
Task Name: mon_config_startup
Task ID: 3
Task Type: STARTUP SENSOR
Last Error
  Number -310
  Message Table (informix.mon_onconfig) already exists in database.
  Time 09/11/2007 11:41
  Task Name mon_config_startup
Task Execution: onconfig_save_diffs

WORKER PROFILE
  Total Jobs Executed 10
  Sensors Executed 8
  Tasks Executed 2
  Purge Requests 8
  Rows Purged 0

Worker Thread(1) 46fa6f80
=====
Task: 4729fc18
Task Name: mon_sysenv
Task ID: 4
Task Type: STARTUP SENSOR
Task Execution: insert into mon_sysenv select 1, env_name, env_value FROM
sysmaster:sysenv

WORKER PROFILE
  Total Jobs Executed 3
  Sensors Executed 2
  Tasks Executed 1
  Purge Requests 2
  Rows Purged 0

Scheduler Thread 46fa6f80
=====
Run Queue
  Empty
Run Queue Size 0
Next Task 7
Next Task Waittime 57
```

Figure 20-28. onstat -g dbc command output

Output description

Worker Thread

Address of the worker thread in shared memory

Task Name of the last executed task

Task ID

The task ID from the **tk_id** column in the **sysadmin:ph_task** table for this task

Task Type

Type of the task

Last Error

Error number, error message, time (in seconds), and task name from the last error the dbWorker thread encountered. It could be from the previously executed task or from a task executed days ago.

Task Execution

SQL statement or SPL procedure or routine executed as part of the task

WORKER PROFILE

The dbWorker thread profile data shows the total jobs executed, number of sensors executed, number of tasks executed, number of purge requests, and the number of rows purged from the result tables for all sensors executed by this dbWorker thread.

Scheduler Thread

Address of the scheduler thread in shared memory

Run Queue

The task ID for the next scheduled task. If no task is scheduled, the value is Empty.

Run Queue Size

The number of tasks that are waiting to be executed by the dbWorker thread

Next Task

The task ID of the next task that will be scheduled to be executed

Next Task Waittime

The number of seconds before the Next Task will be scheduled for execution

Related reference:

 [Monitor the scheduler \(Administrator's Guide\)](#)

“scheduler argument: Stop or start the scheduler (SQL administration API)” on page 21-101

onstat -g ddr command: Print ER database log reader status

Use the **onstat -g ddr** command to display information about the status of the Enterprise Replication database log reader.

Syntax:

►►—onstat— -g—ddr—►►

The **ddr**, or **ddr_snoopy** option, is an internal component of Enterprise Replication that reads the log buffers and passes information to the grouper component.

You can use the information from the **onstat -g ddr** command to monitor the *replay position* in the log file and ensure that the replay position is never overwritten, which can cause loss of data. The replay position is the point from where, if a system failure occurs, Enterprise Replication starts re-reading the log information into the log update buffers. All the transactions generated before this position at all of the target servers have been applied by Enterprise Replication or safely stored in the stable queue space. As messages are acknowledged or stored in the stable queue, the replay position should advance. If you notice that replay position is not advancing, this can mean that the stable queue is full or a remote server is down.

The output from the **onstat -g ddr** command shows you a snapshot of the replay position, the *snoopy position*, and the *current position*. The snoopy position identifies the position of the **ddr_snoopy** thread in the logical logs. The snoopy position is the read position. The **ddr_snoopy** thread has read the log records up until this point. The current position is the position where the server has written its last logical log record.

If log reading is blocked, data might not be replicated until the problem is resolved. If the block is not resolved, the database server might overwrite the read (**ddr_snoopy**) position, which means that data will not be replicated. If this occurs, you must manually resynchronize the source and target databases.

You can enable dynamic log creation by setting the CDR_MAX_DYNAMIC_LOGS configuration parameter in the ONCONFIG file. If the current position reaches the log needs position, instead of going into a blocked state, Enterprise Replication automatically adds another log file. If this option is set, the **onstat -g ddr** command displays information about the number of dynamic log requests made. For more information, see the *IBM Informix Enterprise Replication Guide*.

Example output

The following sample output from the **onstat -g ddr** command shows the replay position, the snoopy position, and the current position highlighted.

DDR -- Running						
# Event	Snoopy	Snoopy	Replay	Replay	Current	Current
Buffers	ID	Position	ID	Position	ID	Position
528	24	165018	24	6a018	24	166000
Log Pages Snooped:						
		From	From	Tossed		
		Cache	Disk	(LBC full)		
		247	111	0		
Total dynamic log requests: 0						
DDR events queue						
Type	TX id	Partnum	Row id			

Figure 20-29. **onstat -g ddr** command output

onstat -g defragment command: Print defragment partition extents

Use the **onstat -g defragment** command to display information about the active requests to defragment partition extents.

Syntax:

►► onstat — -g — defragment ————— ◀◀

Example output

Defrag info							
id	table name	tid	dbnum	partnum	status	substatus	errnum
15	stores_demo:informix.stdtab2	49	2	2097155	SEARCHING_FOR_EXTENT	0	0

Figure 20-30. onstat -g defragment command output

Note: This command displays information about defragment requests that are active. If there are no active defragment requests, only the column headings are returned.

Output description

id The ID of the defragment request.

table name
The fully-qualified name of the table that is being defragmented.

tid The thread ID.

dbnum
The dbspace number that is being defragmented.

partnum
The partition number that is being defragmented.

status

- SEARCHING_FOR_EXTENT
- MERGING_EXTENTS
- DEFRAG_COMPLETED
- DEFRAG_FAILED

substatus
The detailed status number, if any.

errnum
The last error number returned from the defragmentation request.

onstat -g dic command: Print table information

Use the **onstat -g dic** command to display a line of information about each table that is cached in the shared-memory dictionary. If you specify a table name, this command prints internal SQL information about that particular table.

Syntax:

►► onstat — -g — dic ————— ◀◀

Example output

```

Dictionary Cache: Number of lists: 31, Maximum list size: 10
list#  size  refcnt  dirty?  heapptr      table name
-----
  1      3      1      no      14b5d890     wbe@oninit_shm:informix.t0010url
              1      no      14cbb820     wbe@oninit_shm:informix.t9051themeval
              0      no      14b63c20     wbe@oninit_shm:informix.t0060hits

  2      2      0      no      14b97420     wbe@oninit_shm:informix.t0120import
              1      no      14b6c820     wbe@oninit_shm:informix.t9110domain

  3      3      0      no      14bce020     wbe@oninit_shm:informix.t0150url
              0      no      14d3d820     contact@oninit_shm:informix.wbtags
              0      no      14c87420     wbe@oninit_shm:informix.wbtags

  4      1      0      no      14b7a420     drug@oninit_shm:abcdef.product    ..
Total number of dictionary entries: 36

```

Figure 20-31. **onstat -g dic** command output

Output description

list# Data dictionary hash chain ID

size Number of entries in this hash

refcnt Number of SQL statements currently referencing one of the cache entries.

dirty? Whether the entry has been modified since last written to disk.

heapptr

Address for the heap used to store this table

table name

Name of table in cache

For more information on the **onstat -g dic** command, see the *IBM Informix Performance Guide*.

onstat -g dis command: Print database server information

Use the **onstat -g dis** command to display a list of database servers, the status of each server, and information about each server, including the location of the **INFORMIXDIR** directory, **sqlhosts** file, and **ONCONFIG** file. You can use this command in any database server mode, including offline.

Syntax:

```

▶▶—onstat— -g—dis—————▶▶

```

Example output

```

There are 2 servers found
Server       : ol_tuxedo
Server Number : 53
Server Type  : IDS
Server Status : Up
Server Version: IBM Informix Version 11.50.UC1
Shared Memory : 0xa000000
INFORMIXDIR  : /local1/engines/ol_tuxedo/dist
ONCONFIG     : /local1/engines/ol_tuxedo/dist/etc/onconfig.ol_tuxedo
SQLHOSTS     : /local1/engines/ol_tuxedo/dist/etc/sqlhosts
Host         : avocet

Server       : ol_9next
Server Number : 0
Server Type  : IDS
Server Status : Down
Server Version:
Shared Memory : 0
INFORMIXDIR  : /local1/engines/ol_9next/dist
ONCONFIG     :
SQLHOSTS     :
Host         :

```

Figure 20-32. **onstat -g dis** command output

Output description

Server Server name

Server Number
Number of the server.

Server Type
Type of server

Server Status
Up means that the server is online, Down means that the server is offline

Server Version
Version of the server

Shared Memory
Location of the shared memory address

INFORMIXDIR
Location of the **\$INFORMIXDIR/** directory on UNIX and in the **%INFORMIXDIR%** directory on Windows.

ONCONFIG
Location of the ONCONFIG file

SQLHOSTS
Location of the **sqlhosts** file

Host Host name of the server

onstat -g dsk command: Print the progress of the currently running compression operation

Use the **onstat -g dsk** command to print information that shows the progress of currently running compression operations, such as compress, repack, shrink, and uncompress.

Syntax:

►► onstat — -g dsk ◀◀

Example output

Partnum	OP	Rows Processed	Cur Page	Approx Prog	Duration	Approx Remaining	Table
0x00100196	COMPRESS	63253	3515	75%	00:00:03	00:00:01	footab

Figure 20-33. onstat -g dsk command output for a compress operation

Partnum	OP	Rows Processed	Cur Page	Approx Prog	Duration	Approx Remaining	Table
0x00100196	REPACK	22900	4285	28%	00:00:01	00:00:02	footab

Figure 20-34. onstat -g dsk command output for a repack operation

Output description

partnum

Partition number of the table or fragment

OP Compression operation, such as compress, repack, or shrink.

Processed

Number of rows processed so far for the specified operation

Curr Page

The current page number that the server is operating on now

Approx Prog

Percentage of the total operation that has completed

Duration

The number of seconds that have elapsed since the operation started

Approx Remaining

The approximate time that remains before the operation is complete

Table Name of the table

onstat -g dll command: Print dynamic link library file list

Use the **onstat -g dll** command to display a list of and the status of dynamic link library (DLL) files that were loaded.

Syntax:

►► onstat — -g dll ◀◀

Example output

The output displays the names of the library files only one time each process group. The flags indicate if the library was loaded when the server was started.

addr	slot	vp	baseaddr	flags	filename
0x4af55310	15	1	0x2a985e3000	PM	/finance/jeffzhang/mylib.udr
0x4b6f2310		2	0x2a985e3000		
0x4b71b310		3	0x2a985e3000		
0x4c09f310	16	1	0x2a985e3000	M	/deptxyz/udrs/geodetic.bld
0x4c0c0310		2	0x2a985e3000		
0x4c0f1310		3	0x2a985e3000		
0x4c112310	17	1	0x7a138e9000		/home/informix/extend/blade.so
0x4c133310		2	0x3a421e1000		
0x4c133310		3	0x3a421e1000		

Figure 20-35. **onstat -g dll** command output

Output description

addr Address of the DLL file

slot Slot number entry in the library table

vp ID of the virtual processor

baseaddr
Base address of the shared library

flags

- M indicates that the thread calling the UDR can migrate from one CPU virtual processor to another CPU virtual processor.
- P indicates that the shared library was loaded when the database server was started.

filename
Name of the DLL file

onstat -g dmp command: Print raw memory

Use the **onstat -g dmp** command to display information about raw memory at a given address for a number of given bytes.

Syntax:

►► onstat — -g — dmp — address — length — ◄◄

Each address and length must be within the allocated memory shown from **onstat -g seg** output. The address specified can be in decimal or hexadecimal format. Hexadecimal addresses must begin with **0x**. You can specify the address in decimal, but doing so requires converting the memory shown from **onstat -g seg** to decimal before using it as a command line argument.

Example output

```
%onstat -g dmp 0x700000011a19d48 100

address          bytes in mem
0700000011a19d48: 07000000 118e0fa8 07000000 11942b40 .....+@
0700000011a19d58: 07000000 10137120 00000000 00000000 .....q .....
0700000011a19d68: 00000000 00000000 00000000 00000000 .....
0700000011a19d78: 07000000 11a19d48 07000000 11a19d48 .....H .....H
0700000011a19d88: 00000000 00000000 00000000 00000000 .....
0700000011a19d98: *
0700000011a19da8: 00000000 ....
```

Figure 20-36. **onstat -g dmp** command output

Output description

address

Memory address of the raw memory.

bytes in mem

Hexadecimal and ASCII representations of the memory contents.

Output from the command is divided into three columns: memory address, hexadecimal values for the bytes in memory, and the ASCII representation of the bytes in memory. The bytes in memory (middle) section displays the first 16 bytes of memory starting at the address specified on the command line. The third column shows the ASCII representation of the hexadecimal data. Periods are displayed for all hexadecimal values that do not have an ASCII character equivalent. ASCII values are shown in order to make searching for plain text easier.

In the example output shown, the fifth line of data displays zeros and the sixth line contains an asterisk. The asterisk indicates an unknown number of repetitions of the previous line, which means that there is no more data after the fourth line.

onstat -g dri command: Print high-availability data replication information

Use the **onstat -g dri** command to print information about high-availability data replication statistics on the current server.

Use the **onstat -g dri ckpt** command to print information about nonblocking checkpoints in HDR servers.

Syntax:

```
►► onstat -g dri [ckpt] ◄◄
```

Example output

```

Data Replication:
Type           State      Paired server      Last DR CKPT (id/pg)  Supports Proxy Writes
primary        on          BB_1                554 / 558             Y

DRINTERVAL     30
DRTIMEOUT      30
DRAUTO         0
DRLOSTFOUND    /vobs/tristarm/sqlldist/etc/dr.lostfound
DRIDXAUTO      0
ENCRYPT_HDR     0
Backlog        0
Last Send      2011/05/04 00:40:32
Last Receive   2011/05/04 00:40:32
Last Ping      2011/05/04 00:40:18

```

Figure 20-37. **onstat -g dri** command output

```

Data Replication:
Type           State      Paired server      Last DR CKPT (id/pg)  Supports Proxy Writes
primary        on          BB_1                554 / 558             Y

DRINTERVAL     30
DRTIMEOUT      30
DRAUTO         0
DRLOSTFOUND    /vobs/tristarm/sqlldist/etc/dr.lostfound
DRIDXAUTO      0
ENCRYPT_HDR     0

DR Checkpoint processing:
Save State      N
Pages Saved     0
Save Area       none
Received log id, page 17,68
Saved log id, page  0,0
Drain log id, page  0,0
Processed log id, page 17,68
Pending checkpoints 0

```

Figure 20-38. **onstat -g dri ckpt** command output

Output description

Type Current type of server: primary, secondary, or standard

State on or off

Paired server

Name of the primary or secondary server that this server is paired with

Last DR CKPT

Last checkpoint ID and page

Supports Proxy Writes

Displays whether the server is currently configured to allow secondary server updates. **Y** = supports secondary server updates, **N** = does not support secondary server updates.

The second section lists the values of the following configuration parameters in the ONCONFIG file:

- DRINTERVAL
- DRTIMEOUT

- DRAUTO
- DRLOSTFOUND
- DRIDXAUTO
- ENCRYPT_HDR
- Backlog - Number of log pages that have not been sent to the secondary server
- Last Send - The time that the last message was sent to the peer node
- Last Receive - The time that the last message was received from the peer node
- Last Ping - The time of the last ping

The third section displays checkpoint processing statistics:

Save State

Displays **B** (for *buffering*) when the server is adding logs to the staging area, **D** (for *draining*) when the server is removing logs from the staging area, or **N** (for *normal*) when the server is operating normally, meaning that no logs are saved.

Pages Saved

Displays the number of log pages that have been saved in the staging area that have yet to be applied.

Save Area

Displays the location of the staged log files.

Received log id,page

Displays the last log ID and page received from the primary server.

Processed log id,page

Displays the last log ID and page that have been queued to the recovery pipeline.

saved log id,page

Displays the last log ID and page that was stored in the staging area (if stage state is either **B** or **D**).

drain log id,page

Displays the last log ID and page that have been removed from the staging area.

Pending checkpoints

Displays the number of checkpoints that have been staged but have not yet been applied.

Pending ckpt log id, page

Displays the position of any pending checkpoint records.

onstat -g dsc command: Print distribution cache information

Use the **onstat -g dsc** command to display information about distribution cache.

Syntax:

```
▶▶ onstat -g dsc ▶▶
```

Example output

```

Distribution Cache:
  Number of lists          : 31
  PC_POOLSIZE             : 50
  Number of entries        : 0
  Number of entries in use : 0
Distribution Cache Entries:
list#  id  ref_cnt  dropped?  heap_ptr      distribution name
-----
Distribution Cache is empty.

```

Figure 20-39. **onstat -g dsc** command output

Output description

The first section of output describes the distribution cache.

Number of lists

Number of lists in the distribution cache

PC_POOLSIZE

Number of entries that can be cached at one time

Number of entries

Number of entries in the distribution cache

Number of entries in use

Number of entries being used

The second section of output describes the distribution cache entries.

list# Distribution cache hash chain ID

id Number of hash entries

ref_cnt

Number of statements referencing a cache entry

dropped?

Whether this entry has been dropped since being added to the cache

heap_ptr

Heap address used to store this entry

distribution name

The name of the distribution in the cache

onstat -g dss command: Print ER environment data

Use the **onstat -g dss** command to print detailed statistical information about the activity of individual data sync threads in an Enterprise Replication environment.

Syntax:

```

▶▶ onstat -g dss [modifier] ▶▶

```

The data sync thread applies the transaction on the target server. Statistics include the number of applied transactions and failures and when the last transaction from a source was applied.

The **onstat -g dss** command has the following formats:

```
onstat -g dss
onstat -g dss modifier
```

The following table describes the values for *modifier*.

Modifier	Action
UDR	Prints summary information about any UDR invocations by the data sync threads.
UDRx	Prints expanded information (including a summary of error information) about any UDR invocations by the data sync threads. The ProcId column lists the UDR procedure ID.

Example output

In the following example, only one data sync thread is currently processing the replicated data. It has applied a total of one replicated transaction and the transaction was applied at 2004/09/13 18:13:10. The Processed Time field shows the time when the last transaction was processed by this data sync thread.

DS thread statistic						
cmtTime	Tx	Tx	Tx	Last Tx		
Name	< local	Committed	Aborted	Processed	Processed Time	

CDRD_1	0	1	0	1	(1095117190)	2008/09/13 18:13:10
Tables (0.0%):						
Databases: test						
CDR_DSLOCKWAIT = 1						
CDR_DSCLOSEINTERVAL = 60						

Figure 20-40. **onstat -g dss** Output

onstat -g dtc command: Print delete table cleaner statistics

Use the **onstat -g dtc** command to display statistics about the delete table cleaner. The delete table cleaner removes rows from the delete table when the rows are not needed.

The **onstat -g dtc** command is used primarily as a debugging tool and most often by IBM Support.

Syntax:

```
▶▶ onstat -g dtc ◀◀
```

Example output

In the following example, the thread name of the delete table cleaner is **CDRDTCleaner**. The total number of rows deleted is **1**. The last activity on this thread occurred at 2010/09/13 18:47:19. The delete table for replicate **rep1** was last cleaned at 2010/09/13 18:28:25.

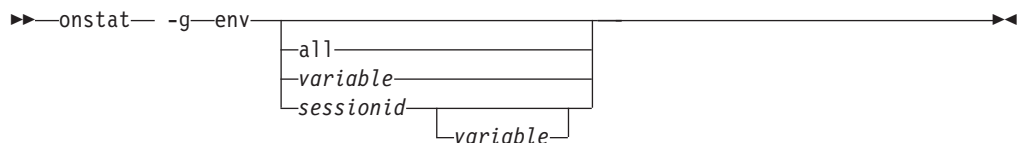
```
-- Delete Table Cleanup Status as of (1095119368) 2010/09/13 18:49:28
  thread          = 49 <CDRDTCleaner>
  rows deleted    = 1
  lock timeouts   = 0
  cleanup interval = 300
  list size       = 3
  last activity    = (1095119239) 2010/09/13 18:47:19
Id      Database      Last Cleanup Time
Replicate      Server      Last Log Change
=====
000001  test              (1095118105) 2010/09/13 18:28:25
      repl              g_bombay      (1095118105) 2010 /09/13 18:28:25
      repl              g_delhi       (1095118105) 2010 /09/13 18:28:25
000002  test              <never cleaned>
```

Figure 20-41. **onstat -g dtc** command output

onstat -g env command: Print environment variable values

Use the **onstat -g env** command to display the values of the environment variables that the database server currently uses.

Syntax:



You can specify one of the following invocations.

Invocation	Explanation
onstat -g env	Displays the settings of environment variables when the database server was started Does not display environment variables that have not been set explicitly.
onstat -g env all	Displays the settings used by all sessions This display is the same as the output of onstat -g env and onstat -g env sessionid iteratively on all current sessions.
onstat -g env variable	Displays the default value of the specified environment variable This <i>variable</i> argument eliminates the need to pipe the output to grep (or some other utility) to locate an environment variable among many that might be set.
onstat -g env sessionid	Displays the settings that a specific session uses. This display includes the following values: <ul style="list-style-type: none"> Set in the environment of the session Assigned by the database server, as onstat -g env displays

Invocation	Explanation
onstat -g env <i>sessionid variable</i>	<p>Displays the value of the specified environment variable that the specified session uses</p> <p>The <i>sessionid</i> and <i>variable</i> arguments eliminate the need to pipe the output to grep (or some other utility) to locate an environment variable among many that might be set.</p>

The **onstat -g env** command displays the current setting of an environment variable and the complete list of values each time the variable was set in the environment. For example, if PDQPRIORITY is set to 10 in the **.informix.rc** file and set to 55 in the shell environment, **onstat -g env** command displays both values.

However, if you change the PDQPRIORITY with the **onmode -q pdqpriority *sessionid*** command, the **onstat -g env** command does not display the new value for the session. The **onstat -g env** command displays only the values of environment variables set in the environment. It does not display values modified while the session is running.

You might want to display the values of environment variables in the following situations:

- The database server instance has been up for months, and you cannot remember the setting of an environment variable (such as the server locale setting **SERVER_LOCALE**).
- You want to display the complete list of values for an environment variable to identify when an environment variable has been set in multiple places.
- Environment files on disk might have changed or been lost in the interim.
- A support engineer wants to know settings of specific environment variables.

Example output

The following figure shows the output for the **onstat -g env** command.

Variable	Value [values-list]
DBDATE	DMY4/
DBDELIMITER	
DBPATH	.
DBPRINT	lp -s
DBTEMP	/tmp
INFORMIXDIR	/build2/11.50/tristarm/sqldist [/build2/11.50/tristarm/sqldist] [/usr/informix]
INFORMIXSERVER	parata1150
INFORMIXTERM	termcap
LANG	C
LC_COLLATE	C
LC_CTYPE	C
LC_MONETARY	C
LC_NUMERIC	C
LC_TIME	C
LD_LIBRARY_PATH	/usr/openwin/lib:/lib:/usr/lib
LKNOTIFY	yes
LOCKDOWN	no
NODEFDAC	no
NON_M6_ATTRS_OK	1
PATH	/build2/11.50/tristarm/sqldist/bin:: /root/bin:/opt/SUNWspro/bin:/usr/ccs/bin: /usr/openwin/bin:/usr/sbin:/usr/bin:/usr /local/binSERVER_LOCALE en_US.819
SHELL	/bin/ksh
SINGLELEVEL	no
SUBQCACHESZ	10
TBCONFIG	onconfig
TERM	xterm [xterm] [dumb]
TERMCAP	/etc/termcap
TZ	GB

Figure 20-42. **onstat -g env** command output

onstat -g ffr command: Print free fragments

Use the **onstat -g ffr** command to display information about the free memory fragments for a specified session or shared-memory pool.

This command requires an additional argument to specify either a pool name or session ID whose memory pool information is to be displayed. Each session is allocated a memory pool with the same name as the session ID. Use the **onstat -g mem** command to identify the pool name and the **onstat -g ses** command to identify the session ID.

Syntax:

```

▶▶ onstat -g ffr [pool name]
                  [sessionid]

```

Example output

```
Free lists for pool name aio:
addr      size    idx
165dcfa0  96       10
1659cf68  152      17
165b2f20  224      26
165c7f20  224      26
1666ec38  968      79
149f2ba0  1120     84
```

Figure 20-43. **onstat -g ffr aio** command output

Output description

addr (hexadecimal)

Memory address of the pool fragment.

size (decimal)

Size, in bytes, of the pool fragment.

idx (decimal)

For internal use. Index in the array of free list pointers.

onstat -g glo command: Print global multithreading information

Use the **onstat -g glo** command to display global information about multithreading, information about each virtual processor that is currently running, and cumulative statistics for each virtual-processor class.

Syntax:

```
▶▶ onstat -g glo ▶▶
```

Example output

```

MT global info:
sessions threads vps lngspins
0 28 11 0

total: sched calls thread switches yield 0 yield n yield forever
per sec: 0 19577737 16256991 3688561 11340793 280099
0 0 0 0 0

Virtual processor summary:
class vps usercpu syscpu total
cpu 1 107.40 8.09 115.49
aio 6 4.19 94.42 98.61
lio 1 0.68 3.54 4.22
pio 1 0.03 0.33 0.36
adm 1 0.00 0.05 0.05
msc 1 0.00 0.01 0.01
total 11 112.30 106.44 218.74

Individual virtual processors:
vp pid class usercpu syscpu total Thread Eff
1 5738 cpu 107.40 8.09 115.49 246.14 46%
2 5739 adm 0.00 0.05 0.05 0.00 0%
3 5740 lio 0.68 3.54 4.22 36.44 11%
4 5741 pio 0.03 0.33 0.36 7.50 4%
5 5742 aio 2.03 51.92 53.95 358.09 15%

```

Figure 20-44. **onstat -g glo** command output

Output description

The following table explains each column in the global information section of the example output.

Table 20-22. Description of the columns in the virtual processor summary

Column name	Description
sessions	The number of sessions
threads	The total number of threads
vps	The total number of virtual processors
lngspins	The number of times a thread had to spin more than 10,000 times to acquire a latch on a resource
sched calls	The total number of scheduled calls.
thread switches	The total number of switches from one thread to another.
yield	Statistics on thread yields, which occur when a thread can no longer continue its task until some condition occurs

The following table explains each column in the virtual processor summary section of the example output.

Table 20-23. Description of the columns in the virtual processor summary

Column name	Description
class	The type of virtual processor.
vps	The number of instances of this class of virtual processor.

Table 20-23. Description of the columns in the virtual processor summary (continued)

Column name	Description
usercpu	The total user time, in seconds, that this class of virtual processor has spent running on the CPU.
syscpu	The total system time, in seconds, this class of virtual processor has spent running on the CPU.
total	The total number of virtual processors, user time, and system time.

The following table explains each column in the individual virtual processors section of the example output.

Table 20-24. Description of the columns for the individual virtual processors

Column name	Description
vp	The virtual processor number.
pid	The Process ID of this oninit process.
class	The type of virtual processor.
usercpu	The total user time, in seconds, that this class of virtual processor has spent running on the CPU.
syscpu	The total system time, in seconds, that this class of virtual processor has spent running on the CPU.
total	The total number of virtual processors, user time, and system time.
Thread	The total time the threads ran on this virtual processor.
Eff	Efficiency. The ratio of the total CPU time to the total time the threads ran on this virtual processor. The total CPU time is the usercpu + syscpu.

onstat -g grp command: Print ER grouper statistics

Use the **onstat -g grp** command to print statistics about the Enterprise Replication grouper.

Syntax:

```

▶▶ onstat -g grp [modifier]

```

The grouper evaluates the log records, rebuilds the individual log records into the original transaction, packages the transaction, and queues the transaction for transmission.

The **-g grp** option is used primarily as a debugging tool and by Technical Support.

The **onstat -g grp** command has the following formats:

```
onstat -g grp
onstat -g grp modifier
```

Valid modifier values

The following table describes the values for *modifier*.

Modifier	Action
A	Prints all the information printed by the G, T, P, E, R, and S modifiers
E	Prints grouper evaluator statistics
Ex	Prints grouper evaluator statistics, expands user-defined routine (UDR) environments
G	Prints grouper general statistics
L	Prints grouper global list
Lx	Prints grouper global list, expands open transactions
M	Prints grouper compression statistics
Mz	Clears grouper compression statistics
P	Prints grouper table partition statistics
pager	Prints grouper paging statistics
R	Prints grouper replicate statistics
S	Prints grouper serial list head (The serial list head is the first transaction in the list, that is, the next transaction that will be placed in the send queue.)
Sl	Prints grouper serial list (The serial list is the list of transactions, in chronological order.)
Sx	Prints grouper serial list, expands open transactions
T	Prints grouper transaction statistics
UDR	Prints summary information about any UDR invocations by the grouper threads
UDRx	Prints expanded information (including a summary of error information) about any UDR invocations by the grouper threads The ProcId column lists the UDR procedure ID.

Example output for the onstat -g grp command

This section contains sample output from various **onstat -g grp *modifier*** commands. The following sample shows output for the **onstat -g grp** command.

```
Grouper at 0xb014018:
Last Idle Time: (1095122236) 2004/09/13 19:37:16
RSAM interface ring buffer size: 528
RSAM interface ring buffer pending entries: 0
Eval thread interface ring buffer size: 48
Eval thread interface ring buffer pending entries: 0
Log update buffers in use: 0
Max log update buffers used at once: 5
Log update buffer memory in use: 0
Max log update buffer memory used at once: 320
Updates from Log: 16
Log update links allocated: 512
Blob links allocated: 0
Conflict Resolution Blocks Allocated: 0
Memory pool cache: Empty
Last Tx to Queuer began : (1095118105) 2004/09/13 18:28:25
Last Tx to Queuer ended : (1095118105) 2004/09/13 18:28:25
Last Tx to Queuer log ID, position: 12,23
Open Tx: 0
Serial Tx: 0
Tx not sent: 0
Tx sent to Queuer: 2
Tx returned from Queuer: 2
Events sent to Queuer: 7
Events returned from Queuer: 7
Total rows sent to Queuer: 2
Open Tx array size: 1024
Table 'tab' at 0xae8ebb0 [ CDRShadow ]
Table 'tab12' at 0xae445e0 [ CDRShadow ]
```

```

Grouper Table Partitions:
Slot 312...
'tab' 1048888
Slot 770...
'tab12' 3145730
Slot 1026...
'tab12' 4194306
Repl links on global free list: 2
Evaluators: 3
Evaluator at 0xb03d030 ID 0 [Idle:Idle] Protection:unused
Eval iteration: 1264
Updates evaluated: 0
Repl links on local free list: 256
UDR environment table at 0xb03d080
Number of environments:      0
Table memory limit   :    25165
Table memory used    :      0
SAPI memory limit    :   131072
SAPI memory used     :      0
Count failed UDR calls:      0
Evaluator at 0xb03d0d8 ID 1 [Idle:Idle] Protection:unused
Eval iteration: 1265
Updates evaluated: 2
Repl links on local free list: 254
UDR environment table at 0xb03d128
Number of environments:      0
Table memory limit   :    25165
Table memory used    :      0
SAPI memory limit    :   131072
SAPI memory used     :      0
Count failed UDR calls:      0
Evaluator at 0xb03d180 ID 2 [Idle:Idle] Protection:unused
Eval iteration: 1266
Updates evaluated: 4
Repl links on local free list: 256
UDR environment table at 0xb03d1d0
Number of environments:      0
Table memory limit   :    25165
Table memory used    :      0
SAPI memory limit    :   131072
SAPI memory used     :      0
Count failed UDR calls:      0
Total Free Repl links 768

```

```

Replication Group 6553601 at 0xb0a8360
Replication at 0xb0a82b0 6553601:6553601 (tab) [ NotifyDS FullRowOn ]
Column Information [ CDRShadow VarUDTs InOrder Same ]
CDR Shadow: offset 0, size 8
In Order: offset 8, size 10
Replication Group 6553602 at 0xb0a8480
Replication at 0xb0a83d0 6553602:6553602 (tab12) [ Ignore Stopped
NotifyDS FullRowOn ]
Column Information [ CDRShadow VarUDTs InOrder Same ]
CDR Shadow: offset 0, size 8
In Order: offset 8, size 16

```

Figure 20-45. **onstat -g grp** command output

Example output for the **onstat -g grp E** command

The following example shows output for the **onstat -g grp E** command. The field **Evaluators: 4** indicates that there are four evaluation threads configured for the

system.

```
Repl links on global free list: 0 Evaluators: 4
Evaluator at 0xba71840 ID 0 [Idle:Idle] Protection: unused
  Eval iteration: 1007
  Updates evaluated: 0
  Repl links on local free list: 256
  UDR environment table at 0xba71890
    Number of environments:      0
    Table memory limit   :      16777
    Table memory used    :          0
    SAPI memory limit    :     131072
    SAPI memory used     :          0
    Count failed UDR calls:      0
Evaluator at 0xba718f0 ID 1 [Idle:Idle] Protection: unused
  Eval iteration: 1007
  Updates evaluated: 0
  Repl links on local free list: 256
  UDR environment table at 0xba71940
    Number of environments:      0
    Table memory limit   :      16777
    Table memory used    :          0
    SAPI memory limit    :     131072
    SAPI memory used     :          0
    Count failed UDR calls:      0
Evaluator at 0xba8c260 ID 2 [Idle:Idle] Protection: unused
  Eval iteration: 1007
  Updates evaluated: 0
  Repl links on local free list: 256
  UDR environment table at 0xba8c2b0
    Number of environments:      0
    Table memory limit   :      16777
    Table memory used    :          0
    SAPI memory limit    :     131072
    SAPI memory used     :          0
    Count failed UDR calls:      0
Evaluator at 0xbaac2a0 ID 3 [Idle:Idle] Protection: unused
  Eval iteration: 1007
  Updates evaluated: 0
  Repl links on local free list: 256
  UDR environment table at 0xbaac2f0
    Number of environments:      0
    Table memory limit   :      16777
    Table memory used    :          0
    SAPI memory limit    :     131072
    SAPI memory used     :          0
    Count failed UDR calls:      0
Total Free Repl links 1024
```

Figure 20-46. **onstat -g grp E** command output

Example output for the **onstat -g grp G** command

The following example shows output for the **onstat -g grp G** command.

```

Grouper at 0xb8ab020:
Last Idle Time: (1095115397) 2004/09/13 17:43:17
RSAM interface ring buffer size: 1040
RSAM interface ring buffer pending entries: 0
Eval thread interface ring buffer size: 64
Eval thread interface ring buffer pending entries: 0
Log update buffers in use: 0
Max log update buffers used at once: 1
Log update buffer memory in use: 0
Max log update buffer memory used at once: 64
Updates from Log: 1
Log update links allocated: 512
Blob links allocated: 0
Conflict Resolution Blocks Allocated: 0
Memory pool cache: Empty

```

Figure 20-47. **onstat -g grp G** command output

Example output for the onstat -g grp P command

The following example shows output for the **onstat -g grp P** command. In the example, the grouper is evaluating rows for the **account**, **teller** and **customer** tables.

```

Table 'teller' at 0xb851480 [ CDRShadow VarChars ]
Table 'account' at 0xb7faad8 [CDRShadow VarChars VarUDTs Floats Blobs]
Table 'customer' at 0xbbe67a8 [CDRShadow VarChars VarUDTs]
Grouper Table Partitions:
  Slot 387...
    'account' 1048707
  Slot 389...
    'teller' 1048709
  Slot 394...
    'customer' 1048714

```

Figure 20-48. **onstat -g grp P** command output

Example output for the onstat -g grp pager command

The following example shows output for the **onstat -g grp pager** command. The sample output shows the grouper large transaction evaluation statistics.

```

Grouper Pager statistics:
Number of active big transactions: 0
Total number of big transactions processed: 0
Spool size of the biggest transaction processed: 0 Bytes

```

Figure 20-49. **onstat -g grp pager** command output

Example output for the onstat -g grp R command

The following example shows output for the **onstat -g grp R** command. In this example, the grouper is configured to evaluate rows for replicates with IDs **6553601** and **6553602** (you can use the **onstat -g cat repls** command to obtain the replicate names). The **Ignore** attribute of replicate ID **6553602** shows that the grouper is currently not evaluating rows for this replicate. This can happen if the replicate state is not **ACTIVE**. You can obtain the replicate state using the **onstat -g cat repls** command.

```

Replication Group 6553601 at 0xb0a8360
  Replication at 0xb0a82b0 6553601:6553601 (tab) [ NotifyDS FullRowOn ]
  Column Information [ CDRShadow VarUDTs InOrder Same ]
  CDR Shadow: offset 0, size 8
  In Order: offset 8, size 10
Replication Group 6553602 at 0xb0a8480
  Replication at 0xb0a83d0 6553602:6553602 (tab12) [ Ignore Stopped NotifyDS FullRowOn ]
  Column Information [ CDRShadow VarUDTs InOrder Same ]
  CDR Shadow: offset 0, size 8
  In Order: offset 8, size 16

```

Figure 20-50. **onstat -g grp R** command output

Example output for the **onstat -g grp T** command

The following example shows output for the **onstat -g grp T** command. In this example, the grouper evaluated and queued 1 transaction to the send queue. The **Tx sent to Queuer** field shows the total number of transactions evaluated and queued to the send queue for propagating to all the replicate participants. The **Total rows sent to Queuer** field shows the total number of rows queued to the send queue for propagating to all the replicate participants.

```

Last Tx to Queuer began : (1095116676) 2004/09/13 18:04:36
Last Tx to Queuer ended : (1095116676) 2004/09/13 18:04:36
Last Tx to Queuer log ID, position: 5,3236032
Open Tx: 0
Serial Tx: 0
Tx not sent: 0
Tx sent to Queuer: 1
Tx returned from Queuer: 0
Events sent to Queuer: 0
Events returned from Queuer: 0
Total rows sent to Queuer: 1
Open Tx array size: 1024

```

Figure 20-51. **onstat -g grp T** command output

onstat -g his command: Print SQL trace information

Use the **onstat -g his** command to display SQL trace information from the collection of **syssqltrace** tables (**syssqltrace**, **syssqltrace_info**, **syssqltrace_hvar** and **syssqltrace_itr**) in the **sysmaster** database.

The **level** setting of the SQLTRACE configuration parameter affects what SQL trace information is stored and displayed by the set of **syssqltrace** tables, and what information **onstat -g his** displays. Each row of the **syssqltrace** table describes a previously executed SQL statement. By default, only the DBSA can view the **syssqltrace** information from the **onstat -g his** command. However, when the **UNSECURE_ONSTAT** configuration parameter is set to 1, all users can view this information.

Syntax:

```

▶▶ onstat — -g — his —————▶▶

```

Example output

The content of the output depends on the trace settings.

Figure 20-52. **onstat -g his** command output

The **Statement history** section in the output provides information about the current settings for tracing.

Statement history:

```
Trace Level           Low
Trace Mode            Global
Number of traces      1000
Current Stmt ID       2
Trace Buffer size      2008
Duration of buffer    293 Seconds
Trace Flags           0x00001611
Control Block         0x4c2f0028
```

The following table describes this output:

Information	Description
Trace Level	Amount of information traced. Valid values are LOW, MED, HIGH, and OFF.
Trace Mode	Type of tracing performed. Global refers to all users on the system User refers to only those users who have tracing enabled by an SQL administration API function.
Number of traces	The number of SQL statements that are being traced. This is the value set in your onconfig file unless the ntraces parameter is changed dynamically through SQL Administration API functions. The range is 500 to 2147483647. If you have 100,000 trace buffers and your organization runs 1000 SQL statements a second, and are tracing all of the statements, then the buffers would last for 100 seconds before they would begin being overwritten.
Current Stmt ID	The ID for the current SQL statement. Each statement being traced gets a unique ID.
Trace Buffer size	The amount of data each trace buffer will capture, in bytes. If you set the size to 2KB, but have an SQL statement that is 12KB, the statement is truncated by at least 10KB. More data might be truncated, depending on what else is being traced.
Duration of buffer	The amount of time, in seconds, that the trace data in the current trace buffer spans. This is not how long the sqltrace feature has been running. In the above example Duration of buffer is 293 seconds which indicates the number of seconds between the first and last SQL statement that are traced.
Trace Flags	The current SQL trace flags that are set.

Information	Description
Control Block	The memory address of the SQL trace control block.

The information displayed below is repeated one time for each time a statement was run. In this example there are two variables being called.

Statement # 2: @ 0x4c2f3028

Database: sysmaster

Statement text:
select count(*) from systables,syscolumns where systables.tabid > ? and
systables.nrows < ?

SELECT using tables [systables syscolumns]

The following table describes this output:

Information	Description
Database	The name of the database or part number of the systables entry for the database.
Statement text	The statement text for this SQL statement. If the statement is a stored procedure, then the statement text would display the procedure stack trace. The statement text might be truncated if the statement and the numeric statistics are larger than the trace buffer.

Iterator/Explain

=====

ID	Left	Right	Est Cost	Est Rows	Num Rows	Partnum	Type
3	0	0	17	42	146	1048579	Index Scan
4	0	0	5249	2366	2366	1048580	Seq Scan
2	3	4	5266	99372	345436	0	Nested Join
1	2	0	1	1	1	0	Group

The following table describes this output:

Information	Description
ID	SQL iterator ID
Left	ID of the left input to the iterator
Right	ID of the right input to the iterator
Est Cost	Estimated cost of this iterator
Est Rows	Estimated rows for this iterator
Num Rows	Actual number of rows for this iterator
Partnum	The table or index partition number.
Type	Type of operation

If the SQL statement contains one or more variables, and you are tracing host variables, the **Host Variables** section is included in the output.

Host Variables

=====

1	integer	100
2	float	1000.0000000000000000

The following table describes this output:

Information	Description
Column 1	The position of the variable in the statement.
Column 2	The data type of the variable.
Column 3	The value of the variable.

Statement information:

Sess_id	User_id	Stmt Type	Finish Time	Run Time
5	2053	SELECT	01:08:48	0.4247

The following table describes this output:

Information	Description
Sess_id	The session ID
User_id	The operating system user ID
Stmt Type	The type of SQL statement
Finish Time	The time of day that the SQL statement finished
Run Time	The total amount of time consumed by the virtual processors or threads used to process the statement. For example, if the Finish Time is 1:15:00 and the Run Time is 9 minutes and the start time is not necessarily 1:06:00. There might be multiple virtual processors or threads involved in processing parts of the statement in parallel.

The **Statement Statistics** section in the output provides specific information about the statement.

Statement Statistics:

Page Read	Buffer Read	Read % Cache	Buffer IDX Read	Page Write	Buffer Write	Write % Cache
1285	19444	93.39	0	810	17046	95.25

Lock Requests	Lock Waits	LK Wait Time (S)	Log Space	Num Sorts	Disk Sorts	Memory Sorts
10603	0	0.0000	60.4 KB	0	0	0

Total Executions	Total Time (S)	Avg Time (S)	Max Time (S)	Avg IO Wait	I/O Wait Time (S)	Avg Rows Per Sec
1	30.8660	30.8660	30.8660	0.0141	29.2329	169.8959

Estimated Cost	Estimated Rows	Actual Rows	SQL Error	ISAM Error	Isolation Level	SQL Memory
102	1376	5244	0	0	CR	32608

Information	Description
Page Read	Number of pages that have been read from disk for this SQL statement
Bugger Read	Number of times a page has been read from the buffer pool and not read from disk for this SQL statement
Read % Cache	Percentage of times the page was read from the buffer pool

Information	Description
Buffer IDX Read	This Currently not implemented
Page Write	Number of pages written to disk
Buffer Write	Number of pages modified and sent back to the buffer pool
Write % Cache	Percentage of time that a page was written to the buffer pool but not to disk
Lock Requests	Total number of locks required by this statement
Lock Waits	Number of times this SQL statement waited on locks
LK Wait Time (S)	Amount of time the statement waited for application locks, in seconds
Log Space	Amount of storage space that the SQL statement used in the logical log
Num Sorts	Total number of sorts used to execute the statement
Disk Sorts	Number of sorts which required disk space to execute the sort for this SQL statement
Memory Sorts	Number of sorts executed which executed entirely in memory for this SQL statement
Total Executions	Total number of times this prepared statement has been executed, or the number of times this cursor has been re-used
Total Time (S)	Total time this prepared statement ran, in seconds
Avg Time (S)	Average time this prepared statement required to execute, in seconds
Max Time (S)	Total time to run the prepared SQL statement, in seconds, excluding any time taken by the application. If you prepare a query then run the query 5 times, each time the query is run a trace is added to the trace buffer. The Max Time is the maximum time any one execution took.
Avg IO Wait	Average amount of time the statement waited for I/O, excluding any asynchronous I/O
I/O Wait Time (S)	Amount of time the statement waited for I/O, excluding any asynchronous I/O, in seconds
Avg Rows Per Sec	Average number of rows a second produced by this statement
Estimated Cost	The query optimizer cost associated with the SQL statement
Estimated Rows	Number of rows returned by the statement, as estimated by the query optimizer
Actual Rows	Number of rows returned for this statement
SQL Error	The SQL error number
ISAM Error	The RSAM or ISAM error number

Information	Description
Isolation Level	Isolation level this statement was run with
SQL Memory	Number of bytes this SQL statement required

For the complete schema of the **syssqltrace** System Monitoring Interface table, see “syssqltrace” on page 2-38.

For details of setting the SQLTRACE configuration parameter, see “SQLTRACE Configuration Parameter” on page 1-139.

onstat -g ioa command: Print combined onstat -g information

Use the **onstat -g ioa** command to display combined information from the **onstat -g iob**, **onstat -g ioq**, and **onstat -g iov** commands.

Syntax:

```
▶▶ onstat -g ioa ▶▶
```

Example output

```

AIO global info:
  7 aio es
  4 open files
  64 max global files
32768 max files from setrlimit
AIO I/O queues:
q name/id    len maxlen totalops dskread dskwrite dskcopy
fifo 0      0      0      0      0      0      0
adt 0      0      0      0      0      0      0
msc 0      0      0      0      0      0      0
aio 0      0      0      0      0      0      0
pio 0      0      0      0      0      0      0
lio 0      0      0      0      0      0      0
gfd 3      0      1     607      0     607      0
AIO I/O vps:
/vp s io/s totalops dskread dskwrite dskcopy wakeups io/wup polltries
pollfound kaio_pend
fifo 0 i 0.0 0      0      0      0      0      0.0 0
0      0
fifo 1 i 0.0 0      0      0      0      0      0.0 0
0      0
msc 0 i 0.0 0      0      0      0      0      0.0 0
0      0
aio 0 i 0.3 607    0     607    0     607    1.0 0
0      0
AIO global files:
gfd pathname          totalops dskread dskwrite io/s
3 rootdbs.1          607      0     607    0.3
AIO big buffer usage summary:
      reads
      pages  ops  pgs/op  holes  hl-ops  hls/op
fifo    0    0   0.00    0      0   0.00
kio     0    0   0.00    0      0   0.00
adt     0    0   0.00    0      0   0.00
msc     0    0   0.00    0      0   0.00
aio     0    0   0.00    0      0   0.00
pio     0    0   0.00    0      0   0.00
lio     0    0   0.00    0      0   0.00
      writes
      pages  ops  pgs/op
fifo    0    0   0.00
kio     0    0   0.00
adt     0    0   0.00
msc     0    0   0.00
aio    607  607   1.00
pio     0    0   0.00
lio     0    0   0.00

```

Figure 20-53. **onstat -g ioa** command output

Output description

For a description of each output column, see the individual “**onstat -g iob** command: Print big buffer use summary,” “**onstat -g ioq** command: Print I/O queue information” on page 20-105, and “**onstat -g iov** command: Print AIO VP statistics” on page 20-107 commands.

onstat -g iob command: Print big buffer use summary

Use the **onstat -g iob** command to display a summary of big buffer use.

Syntax:

```

▶▶ onstat — -g iob —▶▶

```

Example output

```

AIO big buffer usage summary:
      reads
      pages  ops  pgs/op  holes  hl-ops  hls/op
fifo      0      0  0.00      0      0  0.00
kio      0      0  0.00      0      0  0.00
adt      0      0  0.00      0      0  0.00
msc      0      0  0.00      0      0  0.00
aio      0      0  0.00      0      0  0.00
pio      0      0  0.00      0      0  0.00
lio      0      0  0.00      0      0  0.00
      writes
      pages  ops  pgs/op
fifo      0      0  0.00
kio      0      0  0.00
adt      0      0  0.00
msc      0      0  0.00
aio     607    607  1.00
pio      0      0  0.00
lio      0      0  0.00

```

Figure 20-54. **onstat -g iob** command output

onstat -g iof command: Print asynchronous I/O statistics

Use the **onstat -g iof** command to display the asynchronous I/O statistics by chunk or file.

This command is similar to the **onstat -D** command, except that **onstat -g iof** also displays information on nonchunk files. It includes information about temporary files and sort-work files.

Syntax:

```

▶▶ onstat -g iof ◀◀

```

Example output

```

AIO global files:
gfd pathname      bytes read  page reads  bytes write  page writes  io/s
3  rootdbs        1918976    937         145061888    70831        36.5

      op type      count      avg. time
      seeks        0          N/A
      reads        937         0.0010
      writes       4088        0.0335
      kaio_reads    0          N/A
      kaio_writes   0          N/A

```

Figure 20-55. **onstat -g iof** command output

Output description

gfd Global file descriptor number for this chunk or file.

pathname
The pathname of the chunk or file.

bytes read
Number of byte reads that have occurred against the chunk or file.

page reads
Number of page reads that have occurred against the chunk or file.

bytes write
Number of byte writes that have occurred against the chunk or file.

page writes

Number of page writes that have occurred against the chunk or file.

io/s

Number of I/O operations that can be performed per second. This value represents the I/O performance of the chunk or file.

op type

Type of operation.

count

Number of times the operation occurred.

avg time

Average time the operation took to complete.

onstat -g iog command: Print AIO global information

Use the **onstat -g iog** command to display global information about AIO.

Syntax:

```
▶▶ onstat -g iog ◀◀
```

Example output

```
AIO global info:
  8 aio es
  5 open files
 64 max global files
```

Figure 20-56. **onstat -g iog** command output

onstat -g ioq command: Print I/O queue information

Use the **onstat -g ioq** command to display statistics about the number and types of operations performed by I/O queues.

Syntax:

```
▶▶ onstat -g ioq [queue_name] ◀◀
```

If a *queue_name* is given then only queues with that name are shown. If no *queue_name* is given then information is given for all queues.

Example output

```

AIO I/O queues:
q name/id      len maxlen totalops  dskread dskwrite  dskcopy
sqli_dbg      0      0      0      0      0      0
fifo          0      0      0      0      0      0
adt           0      0      0      0      0      0
msc           0      1     537      0      0      0
aio           0      3    6537     238    5777      0
pio           0      2    1103      0    1102      0
lio           0      2   11795      0   11794      0
gfd           3      0    17489    1526   15963      0
gfd           4      0   18347    2384   15963      0
gfd           5      0     220      41     179      0
gfd           6      0      4      0      4      0
gfd           7      0      4      0      4      0
gfd           8      0      4      0      4      0
gfd           9      0      9      24      30      0
gfd          10      0     149      40     109      0
gfd          11      0     621     128     493      0
gfd          12      0    1953    1146     807      0
gfd          13      0     409      71     338      0
gfd          14      0     378      60     318      0

```

Figure 20-57. **onstat -g ioq** command output

Output description

q name/id

The name and number of the I/O queue. The name indicates what type of queue it is. The number is used to tell queues of the same name apart.

Here is a list of the possible queue names and what each type of queue handles:

sqli_dbg

Handles I/O for IBM Technical Support's SQL Interface Debugging feature

fifo Handles I/O for FIFO VPs

adt Handles auditing I/O

msc Handles miscellaneous I/O

aio Handles IBM Informix asynchronous I/O

kio Handles kernel AIO

pio Handles physical logging I/O

lio Handles logical logging I/O

gfd Global File Descriptor - Each primary and mirror chunk is given a separate global file descriptor. Individual gfd queues are used depending on whether kaio is on and the associated chunk is cooked or raw.

len The number of pending I/O requests in the queue

maxlen

The largest number of I/O requests that have been in the queue at the same time

totalops

The total number of I/O operations that have been completed for the queue

dskread

Total number of completed read operations for the queue

dskwrite

Total number of completed write operations for the queue

dskcopy

Total number of completed copy operations for the queue

onstat -g ipl command: Print index page logging status information

Use the **onstat -g ipl** command to display information about the status of index page logging.

Syntax:

►►—onstat— -g—ipl—◄◄

Example output

```
Index page logging status: Enabled
Index page logging was enabled at: 2008/12/20 16:01:02
```

Figure 20-58. onstat -g ipl command output

Output description

Index page logging status

Status of index page logging: Enabled or Disabled.

Index page logging was enabled at

The date and time at which index page logging was enabled.

onstat -g iov command: Print AIO VP statistics

Use the **onstat -g iov** command to display asynchronous I/O statistics for each virtual processor.

Syntax:

►►—onstat— -g—iov—◄◄

Example output

```

AIO I/O vps:
class/vp/id s io/s totalops dskread dskwrite dskcopy wakeups io/wup errors tempops
fifo 7 0 i 0.0 0 0 0 0 1 0.0 0 0
msc 6 0 i 0.1 9988 0 0 0 7833 1.3 0 9988
aio 5 0 i 0.0 4894 3341 1426 0 4393 1.1 0 0
aio 9 1 i 0.0 41 0 41 0 33 1.2 0 0
pio 4 0 i 0.0 199 0 199 0 200 1.0 0 199
lio 3 0 i 0.0 6344 0 6344 0 6344 1.0 0 6344

```

Figure 20-59. **onstat -g iov** command output

Output description

class The class of the virtual processor.

vp The ID number of the virtual processor within its class.

s Current status of the AIO virtual processor

f Fork

i Idle

s Search

b Busy

o Open

c Close

io/s The average I/O speed (measured in operations per second) for the virtual processor since the time the database server started or since the **onstat -z** command was last run, whichever happened last.

totalops

Total number of I/O operations performed by this virtual processor since the time the database server started or since the **onstat -z** command was last run, whichever happened last.

dskread

Total number of read operations performed by this virtual processor since the time the database server started or since the **onstat -z** command was last run, whichever happened last.

dskwrite

Total number of write operations performed by this virtual processor since the time the database server started or since the **onstat -z** command was last run, whichever happened last.

dskcopy

Total number of copy operations performed by this virtual processor since the time the database server started or since the **onstat -z** command was last run, whichever happened last.

wakeups

For AIO VPs, the number of times the virtual processor has gone idle since the time the database server started or since the **onstat -z** command was last run, whichever happened last.

io/wup

For AIO VPs, the average number of I/O operations performed per wake-up by this virtual processor since the time the database server started or since the **onstat -z** command was last run, whichever happened last.

errors Total number of KAIO out of resource errors.

tempops (decimal)

For internal use only. This is I/O operation counter that is maintained to determine when a new AIO VP should be added. It is applicable only when the AUTO_AIOVPS configuration parameter is enabled.

onstat -g lap command: Print light appends status information

Use the **onstat -g lap** command to display information about the status of light appends occurring in the system.

Syntax:

►►—onstat— -g—lap—►►

Example output

Light Append Info						
session id	address	cur_ppage	la_npusd	la_ndata	la_nrows	bufcnt
31	b60a5e8	ffbff494	2938	2937	93990	4

Figure 20-60. **onstat -g lap** command output

Output description

Session id (decimal)

Session ID performing the light append operation

address (hexadecimal)

Address of the light append buffer

cur_ppage (hexadecimal)

Current physical page address

la_npusd (decimal)

Number of pages allocated

la_ndata (decimal)

Number of data pages appended

la_nrows (decimal)

Number of rows appended

bufcnt (decimal)

Number of light append buffers

onstat -g laq command: Print secondary server queues

Use the **onstat -g laq** command to print information about queues on the secondary server that apply log information received from the primary server.

In a high-availability cluster, the primary server sends log records to one or more secondary servers over the network. Each secondary server continuously replays the transaction logs from the primary server to ensure that data is replicated on the secondary server. Each tblspace on the primary server is assigned a queue on the secondary server in which to receive log records. A thread, called an *apply thread*, applies the logs stored in the queue to the secondary server. The logs are applied in the order in which they were received.

You use the **onstat -g laq** command to monitor the performance of the queues on the secondary server. Use this command if you suspect that the primary server performance is slowed because logs are not replaying quickly enough on the secondary server. The Avg Depth (average depth) column indicates the average number of logs in the queue at the time that the last queue insert operation occurred.

The **onstat -g laq** command is valid only on secondary servers. Running the command on the primary server returns only the **onstat** header output.

Syntax:

```
►►onstat— -g— laq —————►►
```

Example output

Log Apply Info:			
Thread	Queue Size	Total Queued	Avg Depth
xchg_1.0	0	9	5.00
xchg_1.1	0	0	0.00
xchg_1.2	0	7	3.43
xchg_1.3	0	8	1.12
xchg_1.4	0	4	1.50
xchg_1.5	0	4	1.50
xchg_1.6	0	6	1.33
xchg_1.7	0	47	3.11
xchg_1.8	0	37	9.46
xchg_1.9	0	13	3.69
Secondary Apply Queue:		Total Buffers:12 Size:64K Free Buffers:11	
Log Recovery Queue:		Total Buffers:4 Size:32K Free Buffers:2	
Log Page Queue:		Total Buffers:32 Size:2K Free Buffers:32	
Log Record Queue:		Total Buffers:50 Size:64K Free Buffers:49	

Figure 20-61. **onstat -g laq** command output

Output description

- Thread**
The name of the apply thread assigned to receive the log record.
- Queue Size**
The number of log records queued for the apply thread.
- Total Queued**
The total number of queued log records for a given apply thread.
- Avg Depth**
The average number of logs in the queue at the time that the last queue insert operation occurred.
- Secondary Apply Queue**
The secondary apply queue receives log buffers from the primary server. The values displayed represent the total number of buffers allocated to receiving log buffer records, the size of the queue, and the number of unused buffers.
- Log Recovery Queue**
The log recovery queue receives output from the secondary apply queue.

The log buffers are converted to a format compatible with the **ontape** utility. The values displayed represent the total number of stream buffers in the recovery queue, the size of the stream buffers, and the number of unused buffers.

Log Page Queue

The log page queue receives output from the log recovery queue. The **ontape** formatting is removed and the data is divided into individual log pages. The values displayed represent the total number of log pages in the queue, the size of the queue, and the number of unused buffers.

Log Record Queue

The log record queue receives output from the log page queue. The log pages are divided into individual log records. The values displayed represent the total number of log records in the recovery queue, the size of the queue, and the number of unused buffers.

onstat -g lmm command: Print low memory management information

Use the **onstat -g lmm** command to display information about automatic low memory management settings and recent activity.

Syntax:

►►—onstat— -g lmm—◄◄

Example output

Low Memory Manager

Control Block 0x4cfca220
Memory Limit 300000 KB
Used 149952 KB
Start Threshold 10240 KB
Stop Threshold 10 MB
Idle Time 300 Sec
Internal Task Yes
Task Name 'Low Memory Manager'
Low Mem TID 0x4cfd7178
Extra Segments 0

Low Memory Manager Tasks

Task	Count	Last Run
Kill User Sessions	267	04/04/2011.16:57
Kill All Sessions	1	04/04/2011.16:58
Reconfig(reduce)	1	04/04/2011.16:59
Reconfig(restore)	1	04/04/2011.17:59

Last 20 Sessions Killed

Ses ID	Username	Hostname	PID	Time
194	sfisher	host01	13433	04/04/2011.16:57
201	sfisher	host01	13394	04/04/2011.16:57
198	sfisher	host01	13419	04/04/2011.16:57
190	sfisher	host01	13402	04/04/2011.16:57
199	sfisher	host01	13431	04/04/2011.16:57

Total Killed 177

Figure 20-62. **onstat -g lmm** command output

Output description

Control Block

Address of the internal control structure for automatic low memory management

Memory Limit

Amount of memory to which the server is attempting to adhere

Used Amount of memory currently used by the server

Start Threshold

Value for the automatic low memory management start threshold

Stop Threshold

Value for the automatic low memory management stop threshold

Idle Time

The amount of time after which automatic low memory management considers a session idle

Internal Task

Yes = using Informix procedures

No = using user-defined procedures

Task Name

Name of user-defined procedure

Low Mem TID

Address of the automatic low memory management thread

Task Kill = Automatic processes ran and terminated sessions.
 Reconfig(reduce) = Automatic processes ran and freed blocks of unused memory.
 Reconfig(restore) = Automatic processes ran and restored services and configuration.

Count Number of times that the task ran

Last Run
 Date and time when the last task ran

Ses ID
 ID of session that was terminated (with an **onmode -z** command)

Username
 User name of the owner of the session

Hostname
 Name of the host where the session originated

PID Process ID

Time Date and time when the session was terminated

You use the LOW_MEMORY_MGR configuration parameter to enable the automatic low memory management.

Related reference:

“scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)” on page 21-101

“scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)” on page 21-104

onstat -g lmx command: Print all locked mutexes

Use the **onstat -g lmx** command to display information about all locked mutexes.

Syntax:

►►—onstat— -g—lmx—◄◄

Example output

```
Locked mutexes:
mid      addr          name                holder  lkcnt  waiter  waittime
Number of mutexes on VP free lists: 49
```

Figure 20-63. **onstat -g lmx** command output

Output description

mid Internal mutex identifier
addr Address of locked mutex
name Name of the mutex
holder Session ID of the thread holding the mutex
lkcnt Number of waiters for this mutex

waiter List of addresses waiting for this mutex

waittime
Amount of time this thread has been waiting

onstat -g lsc command: Print active light scan status (deprecated)

The **onstat -g lsc** command has been superseded by the **onstat -g scn** command.

Syntax:

►► onstat — -g — lsc ————— ◀◀

Example output

Light Scan Info						
descriptor	address	next_lpage	next_ppage	ppage_left	bufcnt	look_aside
3	474b74b0	4a0	7e2c80	416	1	N

Figure 20-64. **onstat -g lsc** command output

Output description

- descriptor (decimal)**
Light scan ID
- address (hex)**
Memory address of the light scan descriptor
- next_lpage (hex)**
Next logical page address to scan
- next_ppage (hex)**
Next physical page address to scan
- ppage_left (decimal)**
Number of physical pages left to scan in the current extent
- #bufcnt (decimal)**
Number of light scan buffers used for this light scan
- #look_aside (char)**
Whether look aside is needed for this light scan (Y = yes, N = no). Look asides occur when a thread needs to examine the buffer pool for existing pages to obtain the latest image of a page being light scanned.

Use the **onstat -g scn** command to display the status of a current scan, based on rows scanned on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data. For more information, see “**onstat -g scn** command: Print scan information” on page 20-155.

onstat -g mem command: Print pool memory statistics

Use the **onstat -g mem** command to display the memory statistics for a pool.

Syntax:

```
▶▶ onstat -g mem [pool name—session id] ▶▶
```

Session pools are named with the session number. If no argument is provided, information about all pools is displayed.

Example output

```
Pool Summary:
name          addr          totalsize freesize #allocfrag #freefrag
resident      R      10a001028      2420736   7960      2          2
res-buff       R      10a250028      8269824   7960      2          2
global        V      10aac0028      9351168  32648     650        11
...
...
...
onmode_mon     V      10b983028       20480    2752     108         1
13             V      10bd5d028       16384    5200      12         2
Blkpool Summary:
name          addr          size      #blks      pre-hint  szavail|
global        V      10aac8920        0         0         0         0
xmf_msc_pl     V      10ac84ca0     954368     73         0         0
```

Figure 20-65. `onstat -g mem` command output

Output description

Pool Summary

name Pool name

Shared memory segment type where the pool is created

addr Pool memory address

totalsize

Pool size, in bytes

freesize

Free memory in pool

#allocfrag

Allocated fragments in pool

#freefrag

Free fragments in pool

Blkpool Summary

name Pool name

Shared memory segment type where pool is created

addr Pool memory address

size Pool size, in bytes

#blks Number of blocks in pool

onstat -g mgm command: Print MGM resource information

Use the **onstat -g mgm** command to display resource information about Memory Grant Manager (MGM).

You can use the **onstat -g mgm** command to monitor how MGM coordinates memory use and scan threads. This command reads shared-memory structures and provides statistics that are accurate at the instant that the command executes.

Syntax:

►►—onstat— -g—mgm—◄◄

The **onstat -g mgm** output displays a unit of memory called a *quantum*. The *memory quantum* represents a unit of memory, as follows:

memory quantum = DS_TOTAL_MEMORY / DS_MAX_QUERIES

The following calculation shows the memory quantum for the values that the **onstat -g mgm** output displays:

memory quantum = 4000 kilobytes / 31
 = 129 kilobytes

The *scan thread quantum* is always equal to 1.

Example output

Memory Grant Manager (MGM)						

MAX_PDQPRIORITY: 100						
DS_MAX_QUERIES: 31						
DS_MAX_SCANS: 1048576						
DS_NONPDQ_QUERY_MEM: 128 KB						
DS_TOTAL_MEMORY: 4000 KB						
Queries:	Active	Ready	Maximum			
		0	0	31		
Memory:	Total	Free	Quantum			
(KB)	4000	4000	128			
Scans:	Total	Free	Quantum			
	1048576	1048576	1			
Load Control:	(Memory)	(Scans)	(Priority)	(Max Queries)	(Reinit)	
	Gate 1	Gate 2	Gate 3	Gate 4	Gate 5	
(Queue Length)	0	0	0	0	0	
Active Queries: None						
Ready Queries: None						
Free Resource	Average #		Minimum #			

Memory	0.0 +- 0.0		500			
Scans	0.0 +- 0.0		1048576			
Queries	Average #		Maximum #	Total #		

Active	0.0 +- 0.0		0	0		
Ready	0.0 +- 0.0		0	0		
Resource/Lock Cycle Prevention count: 0						

Figure 20-66. `onstat -g mgm` command output

Output description

The first portion of the output shows the values of the PDQ configuration parameters.

The second portion of the output describes MGM internal control information. It includes four groups of information. The first group is **Queries**:

- Active* Number of PDQ queries that are currently executing
- Ready* Number of user queries ready to run but whose execution the database server deferred for load-control reason
- Maximum* Maximum number of queries that the database server permits to be active. Reflects current value of the DS_MAX_QUERIES configuration parameter

The next group is **Memory**:

- Total* Kilobytes of memory available for use by PDQ queries (DS_TOTAL_MEMORY specifies this value.)
- Free* Kilobytes of memory for PDQ queries not currently in use
- Quantum* Kilobytes of memory in a memory quantum

The next group is **Scans**:

Total The total number of scan threads as specified by the DS_MAX_SCANS configuration parameter

Free Number of scan threads currently available for decision-support queries

Quantum
The number of scan threads in a scan-thread quantum

The last group in this portion of the output describes **MGM Load Control**:

Memory
Number of queries that are waiting for memory

Scans Number of queries that are waiting for scans

Priority
Number of queries that are waiting for queries with higher PDQ priority to run

Max Queries
Number of queries that are waiting for a query slot

Reinit Number of queries that are waiting for running queries to complete after an **onmode -M** or **-Q** command

The next portion of the output, **Active Queries**, describes the MGM active and ready queues. This portion of the output shows the number of queries waiting at each gate:

Session The session ID for the session that initiated the query

Query Address of the internal control block associated with the query

Priority
PDQ priority assigned to the query

Thread Thread that registered the query with MGM

Memory
Memory currently granted to the query or memory reserved for the query (Unit is MGM pages, which is 8 kilobytes.)

Scans Number of scan threads currently used by the query or number of scan threads allocated to the query

Gate Gate number at which query is waiting

The next portion of the output, **Free Resource**, provides statistics for MGM free resources. The numbers in this portion and in the final portion reflect statistics since system initialization or the last **onmode -Q**, **-M**, or **-S** command. This portion of the output contains the following information:

Average
Average amount of memory and number of scans

Minimum
Minimum available memory and number of scans

The next portion of the output, **Queries**, provides statistics concerning MGM queries:

Average
Average active and ready queue length

Maximum

Maximum active and ready queue length

Total

Total active and ready queue length

Resource/Lock Cycle Prevention count

Number of times the system immediately activated a query to avoid a potential deadlock. (The database server can detect when some of the queries in its queue might create a deadlock situation if the queries are not run immediately.)

onstat -g nbm command: Print a block bit map

Use the **onstat -g nbm** command to display the block bit map for the nonresident segments.

Each bit of the bitmap represents a 4 KB block. If the block is used, then the bit is set to 1. If the block is free, the bit is set to 0. The bitmap is shown as a series of hexadecimal numbers. The bits, and therefore the blocks, are numbered starting at 0 so the first block is block 0, the second is block 1, and so on.

Syntax:

►►—onstat— -g—nbm—►►

Example output

This example shows the bitmap for the segment of virtual memory at 0x10CC00000. The bitmap itself is at 0x10CC00290. All 1792 blocks of the segment are free except for block 0 and block 1023.

```
Block bitmap for virtual segment address 0x10cc00000:
address = 0x10cc00290, size(bits) = 1792
used = 1, largest_free = -1
  0:8000000000000000 0000000000000000 0000000000000000 0000000000000000
256:0000000000000000 0000000000000000 0000000000000000 0000000000000000
512:0000000000000000 0000000000000000 0000000000000000 0000000000000000
768:0000000000000000 0000000000000000 0000000000000000 0000000000000001
1024:0000000000000000 0000000000000000 0000000000000000 0000000000000000
1280:0000000000000000 0000000000000000 0000000000000000 0000000000000000
1536:0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

Figure 20-67. **onstat -g nbm** command output

Output description

address

The starting address of the bitmap.

size

The number of bits in the bitmap. This is also the number of 4 KB blocks in the memory segment.

used

The total number of bits in the bitmap that are set to 1. This is also the number of 4 KB blocks that are in use in the memory segment.

largest free

If this is a value other than -1 it is the largest number of consecutive bits that are free, which is also the number of 4 KB blocks in the largest contiguous set of blocks in the memory segment.

A value of -1 means that the largest free space has not been calculated. The database server only calculates the largest free space if it tries to allocate a set of blocks starting at the *lastalloc* block but there is not enough free space. The value is set to -1 again as soon as another block is allocated in the segment.

onstat -g nif command: Print statistics about the network interface

Use the **onstat -g nif** command to display statistics about the network interface.

The **onstat -g nif** command is used primarily as a debugging tool , most often by Technical Support.

The output shows which sites are connected and provides a summary of the number of bytes sent and received by each site. This can help you determine if a site is not sending or receiving bytes.

Syntax:



The following table describes the options for the **onstat -g nif** command:

Option	Action
all	Prints the sum and the sites.
sites	Prints the NIF site context blocks.
<i>server_ID</i>	Prints information about the replication server with that server ID.
sum	Prints the sum of the number of buffers sent and received for each site.

Example output

The following example shows output for the **onstat -g nif** command. In this example, the local server is connected to the server group **g_bombay** and its CDR ID is **200**. The connection status is running. The connection between the two servers is running, but the replication state on the **g_bombay** server is suspended. The server group **g_bombay** internal NIF version is **9**. The local server has sent three messages to the server **g_bombay** and it has received two messages from **g_bombay**.

NIF anchor Block: af01610
nifGState RUN
RetryTimeout 300
CDR connections:
Id Name State Version Sent Received

200 g_bombay RUN,SUSPEND 9 3 2

Figure 20-68. `onstat -g nif` command output

Output description

NIF anchor Block

The address of the network storage block.

nifGState

The connection state.

RetryTimeout

The number of seconds before Enterprise Replication attempts to retry a dropped connection.

Id The Enterprise Replication ID number for the server.

Name The name of the server group.

State The connection state between the local server and the listed server. If multiple states are shown the second state designates the replication state.

Version

The internal version number of the NIF component on the listed server.

Sent The number of messages the local server has sent to the listed server.

Received

The number of messages received by the local server from the listed server.

onstat -g nsc command: Print current shared memory connection information

Use the `onstat -g nsc` command to display information about shared memory connections either for all of the current connections or for a specified connection ID.

Syntax:



If no *client_id* is provided, information about all current shared memory connections to the database server is given. If a *client_id* is provided then this command gives more detailed information about the shared memory connection with that ID.

Example output

This is output of `onstat -g nsc` with no *client_id*. It shows that there is only one user currently connecting to the database server through shared memory. That

connection has an ID of 0.

clientid	clientPID	state	#serverbufs	#clientbufs	#rdwrts
0	6031	Connected	4	4	12

Figure 20-69. **onstat -g nsc** command output

This example shows output from running the command using a *client_id* of 0.

Network Shared Memory Status for Client: 0

clientid	clientPID	state	#serverbufs	#clientbufs	#rdwrts
0	18949	Connected	4	4	447048

needbuf	segid	semid	semnum	be_semid	be_semnum
0	1303	851969	0	851969	10

be_curread	be_curwrite	fe_curread	fe_curwrite
-1	1	0	2

be_nextread	be_nextwrite	fe_nextread	fe_nextwrite
2	2	4	3

readyqueue

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
----	----	----	----	----	----	----	----	----	----	----

Server Buffers					Client Buffers				
i:	bufid	status	offset	fe_addr	bufid	status	offset	fe_addr	
0:	4	inuse	4474	804474	0	avail	3424	803424	
1:	5	inuse	4888	804888	1	avail	3838	803838	
2:	6	avail	4c9c	804c9c	2	inuse	3c4c	803c4c	
3:	7	avail	50b0	8050b0	3	avail	4060	804060	
4:	-1	free	0	0	-1	free	0	0	
5:	-1	free	0	0	-1	free	0	0	

Figure 20-70. **onstat -g nsc** command with client id output

Output description

clientid

Server assigned ID

clientPID

Client process ID

state State of connection

Connected

The client has established a connection with the server.

Con1 The server has successfully set up a connection with the client, but the client has not yet been notified of it.

Waiting

The server is in the process of setting up a connection with the client.

Reject Client connection has been rejected by the server, normally because the server is shutting down or not yet in on-line mode.

Closed

Server has closed the connection with the client. Client might not be aware of the fact yet.

Not connected

Server is initializing internal structures for the connection.

Unknown

Connection has been closed and the client is aware of the fact.
Server is cleaning up internal structures.

#serverbufs

Database server buffers currently allocated

#clientbufs

Client buffers currently allocated

#rdwrts

The total number of reads and writes performed through this connection since it was created.

The following items are only in the output if you run the **onstat -g nsc** command with a *client_id*:

needbuf

Indicates if server is waiting for a buffer to be freed

0 False

1 True

segid Shared memory segment ID

semid Semaphore ID

semnum

Semaphore number in the semaphore ID

be_semid

Backend semaphore ID

be_semnum

Backend semaphore number in the semaphore ID

be_curread

ID of backend buffer being read

be_curwrite

ID of backend buffer being written

fe_curread

ID of frontend buffer being read

fe_currwrite

ID of frontend buffer being written

be_nextread

ID of next backend buffer to be read

be_nextwrite

ID of next backend buffer to be written

fe_nextread

ID of next frontend buffer to be read

fe_nextwrite
ID of next frontend buffer to be written

readyqueue
Queue of the shared memory buffer ids

Buffers

i Internal location key of message buffer

bufid Message buffer ID

status Status of message buffer

offset Offset of memory buffer in shared memory segments

fe_addr
Frontend address of message buffer

onstat -g nsd command: Print poll threads shared-memory data

Use the **onstat -g nsd** command to display information about shared-memory data for poll threads.

Syntax:

►► onstat — -g — nsd ————— ◀◀

Example output

```
Network Shared Memory Data for Poll Thread: 0
Free Message Buffer Bitmap
(bitmap address = 10b9eef80, bitmap size 480)
000000010b9eef80:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
000000010b9eefa0:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
Free Message Buffer Status Bitmap
(bitmap address = 10ca0a9b0, bitmap size 50)
000000010ca0a9b0:ffffffff ffffff
Message Buffer Table
bufid  clientid      addr
Message Buffer Status Table
clientid      netscb addr      addr      offset
```

Figure 20-71. **onstat -g nsd** command output

onstat -g nss command: Print shared memory network connections status

Use the **onstat -g nss sessionid** command to display information about the status of the shared memory network connections.

Syntax:

►► onstat — -g — nss ———— [sessionid] ————— ◀◀

If no *sessionid* is provided, a one-line summary for each shared memory connection is listed.

Example output

clientid	clientPID	state	#serverbufs	#clientbufs	#rdwrts
1	14018	Connected	4	4	331
0	12398	Connected	4	4	294
2	14036	Connected	4	4	59

Figure 20-72. **onstat -g nss** command output

Output description

clientid (decimal)

Server assigned value for lookups

clientPID (decimal)

Client process ID

state (string)

Current state of the connection.

- Connected
- Con1
- Waiting
- Reject
- Bedcover
- Closed
- Not connected
- Unknown

#serverbufs (dec)

Number of database server buffers currently allocated

#clientbufs (dec)

Number of client buffers currently allocated

#rdwrts (dec)

Total number of buffers in use

onstat -g ntd command: Print network statistics

Use the **onstat -g ntd** command to display network statistics by service.

Syntax:

►►—onstat— -g—ntd—◄◄

Example output

```

global network information:
#netscb connects      read      write q-limits  q-exceed alloc/max
4/  5      11        0      3546 3549/ 10  10/  0    0/  0

Client Type    Calls    Accepted    Rejected      Read      Write
sqlxec        yes      11          0             3531     3540
srvinfo        yes       0          0              0         0
onspac        yes       0          0              4         9
onlog         yes       0          0              0         0
onparam       yes       0          0              0         0
oncheck       yes       0          0              0         0
onload        yes       0          0              0         0
onunload      yes       0          0              0         0
onmonitor     yes       0          0              0         0
dr_accept     yes       0          0              0         0
cdraccept     no        0          0              0         0
ontape        yes       0          0              0         0
srvstat       yes       0          0              0         0
asfecho       yes       0          0              0         0
listener      yes       0          0             11         0
crsamexec     yes       0          0              0         0
onutil        yes       0          0              0         0
drdaexec      yes       0          0              0         0
smx           yes       0          0              0         0
safe          yes       0          0              0         0
Totals                11          0             3546     3549

```

Figure 20-73. **onstat -g ntd** command output

onstat -g ntm command: Print network mail statistics

Use the **onstat -g ntm** command to display statistics about network mail.

Syntax:

```

▶▶ onstat -g ntm ◀◀

```

Example output

```

global network information:
#netscb connects      read      write q-limits  q-exceed alloc/max
4/  5      11        0      3546 3549/ 10  10/  0    0/  0

Network mailbox information:
box  netscb thread name    max received  in box  max in box full signal
5  f07e8b0 soctcpoll      10        24      0         1         0    yes
6  f0b6ad8 soctcp1st      10         0         0         0         0    no
7  f0e8b18 soctcp1st      10         0         0         0         0    no

```

Figure 20-74. **onstat -g ntm** command output

onstat -g ntt command: Print network user times

Use the **onstat -g ntt** command to display information about network user times.

Syntax:

```
▶▶ onstat -g ntt ◀◀
```

Example output

```
global network information:
#netscb connects      read      write      q-limits  q-exceed alloc/max
3/ 3      0      0      135/ 10    0/ 0    2/ 0
Individual thread network information (times):
netscb  thread name  sid  open      read      write      address
c76ea28  ontape            61   14:34:48  14:34:50  14:34:50
c63e548  tlitcplst         4    14:30:43  14:34:48
c631028  tlitcpoll         3    14:32:32
```

Figure 20-75. **onstat -g ntt** command output

onstat -g ntu command: Print network user statistics

Use the **onstat -g ntu** command to display information about network user statistics.

Syntax:

```
▶▶ onstat -g ntu ◀◀
```

Example output

```
global network information:
#netscb connects      read      write      q-free  q-limits  q-exceed alloc/max
2/ 3      16      2611      2603    1/ 1    135/ 10    0/ 0    1/ 1
Individual thread network information (basic):
netscb type  thread name  sid  fd poll  reads  writes q-nrm q-pvt q-exp
d1769f0 soctcp soctcplst    3    1    5     16      0  0/ 0  0/ 0  0/ 0
d1199f0 soctcp soctcpoll    2    0    5    2595      0  0/ 0  0/ 0  0/ 0
```

Figure 20-76. **onstat -g ntu** command output

onstat -g opn command: Print open partitions

Use the **onstat -g opn** command to display a list of the partitions (tables and indexes), by thread ID, that are currently open in the system.

Use the *thread_id* option to restrict the list to a specified ID.

Syntax:

```
▶▶ onstat -g opn [thread_id] ◀◀
```

Example output

tid	rstcb	isfd	op_mode	op_flags	partnum	ucount	ccount	lockmode
38	0x00000000460db7b0	0	0x00000400	0x00000397	0x001000af	2	2	0
38	0x00000000460db7b0	1	0x00000002	0x00000117	0x001000af	2	2	0
38	0x00000000460db7b0	2	0x00000440	0x00000797	0x0010010c	2	0	0
38	0x00000000460db7b0	3	0x00000400	0x00000407	0x0010010a	2	0	0
38	0x00000000460db7b0	4	0x00000400	0x00000407	0x0010010a	2	0	0
38	0x00000000460db7b0	5	0x00000002	0x00000003	0x00100003	2	2	0
38	0x00000000460db7b0	6	0x00000400	0x00000397	0x00100003	2	2	0
38	0x00000000460db7b0	7	0x00000400	0x00000413	0x0010010f	2	0	0
38	0x00000000460db7b0	8	0x00000440	0x00000797	0x0010010c	2	0	0
38	0x00000000460db7b0	9	0x00000402	0x00000403	0x0010010f	2	0	0
38	0x00000000460db7b0	10	0x00000442	0x00000403	0x00100111	1	0	0
38	0x00000000460db7b0	11	0x00000442	0x00000403	0x00100110	1	0	0
38	0x00000000460db7b0	12	0x00000442	0x00000403	0x00100112	1	0	0
38	0x00000000460db7b0	15	0x00000400	0x00004407	0x00000006	1	0	0
38	0x00000000460db7b0	16	0x00000400	0x00000413	0x00100119	1	0	0
36	0x00000000460dbf98	0	0x00000400	0x00000397	0x001000af	2	2	0
36	0x00000000460dbf98	1	0x00000002	0x00000003	0x001000af	2	2	0
36	0x00000000460dbf98	3	0x00000402	0x00000407	0x0010010a	2	0	0
36	0x00000000460dbf98	4	0x00000400	0x00000413	0x0010010a	2	0	0
36	0x00000000460dbf98	6	0x00000442	0x00000797	0x0010010c	1	0	0
37	0x00000000460dc780	0	0x00000400	0x00000397	0x001000af	2	2	0
37	0x00000000460dc780	1	0x00000002	0x00000117	0x001000af	2	2	0
37	0x00000000460dc780	2	0x00000400	0x00000407	0x0010010a	2	0	0
37	0x00000000460dc780	3	0x00000440	0x00000797	0x0010010c	2	0	0
37	0x00000000460dc780	4	0x00000400	0x00000413	0x0010010f	2	0	0
37	0x00000000460dc780	5	0x00000400	0x00000407	0x0010010a	2	0	0
37	0x00000000460dc780	6	0x00000440	0x00000797	0x0010010c	2	0	0
37	0x00000000460dc780	7	0x00000400	0x00000397	0x00100003	2	2	0
37	0x00000000460dc780	8	0x00000002	0x00000003	0x00100003	2	2	0
37	0x00000000460dc780	9	0x00000442	0x00000403	0x00100111	1	0	0
37	0x00000000460dc780	10	0x00000442	0x00000403	0x00100110	1	0	0
37	0x00000000460dc780	11	0x00000402	0x00000403	0x0010010f	2	0	0
37	0x00000000460dc780	12	0x00000400	0x00000413	0x00100119	1	0	0
37	0x00000000460dc780	13	0x00000442	0x00000403	0x00100112	1	0	0
37	0x00000000460dc780	14	0x00000400	0x00004407	0x00000006	1	0	0

Figure 20-77. **onstat -g opn** command output

Output description

tid (decimal)

Thread ID currently accessing the partition resource (table/index)

rstcb (hexadecimal)

In-memory address of the RSAM thread control block for this thread

isfd (decimal)

ISAM file descriptor associated with the open partition

op_mode (hexadecimal)

Current status of the partition lock mode using a combination of the following hexadecimal values:

0x000000 Open for input only
0x000001 Open for output only
0x000002 Open for input and output
0x000004 System catalog
0x000008 No logical logging
0x000010 Open if not already opened for alter
0x000020 Open all fragments data and index
0x000040 Do not allocate a blob descriptor
0x000080 Open for alter
0x000100 Open all data fragments

- 0x000200 Automatic record lock
- 0x000400 Manual record lock
- 0x000800 Exclusive ISAM file lock
- 0x001000 Ignore dataskip - data cannot be ignored
- 0x002000 Dropping partition - delay file open
- 0x004000 Do not drop blob space blobs when table dropped
(alter fragment)
- 0x010000 Open table for DDL operations
- 0x040000 Do not assert fail if this partnum does not exist
- 0x080000 Include fragments of subtables
- 0x100000 Table created under supertable
- 0x400000 Blob in use by CDR

op_flags (hexadecimal)

Current status of the partition using a combination of the following hexadecimal values:

- 0x0001 Open data structure is in use
- 0x0002 Current position exists
- 0x0004 Current record has been read
- 0x0008 Duplicate created or read
- 0x0010 Skip current record on reverse read
- 0x0020 Shared blob information
- 0x0040 Partition opened for rollback
- 0x0080 Stop key has been set
- 0x0100 No index related read aheads
- 0x0200 isstart called for current stop key
- 0x0400 Pseudo-closed
- 0x0800 Real partition opened for SMI query
- 0x1000 Read ahead of parent node is done
- 0x2000 UDR keys loaded
- 0x4000 Open is for a pseudo table
- 0x8000 End of file encountered when positioning in table

partnum (hexadecimal)

Partition number for the open resource (table/index)

ucount (decimal)

Number of user threads currently accessing this partition

ocount (decimal)

Number of times this partition was opened

lockmode (decimal)

Type of lock being held using one of the following coded values:

- 0 No locks
- 1 Byte lock
- 2 Intent shared lock
- 3 Shared lock
- 4 Shared lock by repeatable read (only on items)
- 5 Update lock
- 6 Update lock by repeatable read (only on items)
- 7 Intent exclusive lock
- 8 Shared, intent exclusive lock
- 9 Exclusive lock
- 10 Exclusive lock by repeatable read (only on items)
- 11 Insertion's repeatable read test lock

onstat -g osi: Print operating system information

Use the **onstat -g osi** command to display information on your operating system resources and parameters, including shared memory and semaphore parameters, the amount of memory currently configured on the computer, and the amount of memory that is unused.

Example Output

The **onstat -g osi** command also displays statistics on the hardware processors on your computer.

Use this command when the server is not online.

```
Machine Configuration....
OS Name                      Linux
OS Release                   2.6.9-34.ELsmp
OS Node Name                  idas
OS Version                    #1 SMP
OS Machine                    x86_64
Number of processors          4
Number of online processors   4
System memory page size      4096 bytes
System memory                 7970 MB
System free memory            1536 MB
Number of open files per process 1024
shmmax                        33554432
shmmmin                       1
shmids                        4096
shmNumSegs                    2097152
semmap                        << Unsupported >>
semids                        128
semnum                        32000
semundo                       << Unsupported >>
semNumPerID                   250
semops                        32
semUndoPerProc                << Unsupported >>
semUndoSize                   20
semMaxValue                   32767
```

Figure 20-78. **onstat -g osi** Command Output

onstat -g pos command: Print file values

Use the **onstat -g pos** command to display the values in the `$INFORMIXDIR/etc/.infos.DBSERVERNAME` file.

Syntax:

►► onstat — -g — pos ————— ◀◀

Example output

```
1 7 0 infos ver/size 3 264
2 1 0 snum 0 52564801 44000000 4139 demo_on
3 4 0 onconfig path /opt/IBM/informix/etc/onconfig.demo_on
4 5 0 host informixva
5 6 0 oninit ver IBM Informix Dynamic Server Version 11.70.UC2DE
6 8 0 sqlhosts path /data/IBM/informix/etc/sqlhosts.demos
7 3 -32767 sema 32769
8 2 -32768 shm 32768 52564801 44000000 114176000 R
9 2 1 shm 1 52564802 4ace3000 67108864 V
```

Figure 20-79. **onstat -g pos** command output

onstat -g ppd command: Print partition compression dictionary information

Use the **onstat -g ppd** command to display information about the active compression dictionaries that were created for compressed tables and table fragments. You can choose to print information for a particular numbered partition or for all open partitions.

The **onstat -g ppd** command prints the same information that the **syscompdicts_full** table and the **syscompdicts** view in the **sysmaster** database display. The only difference is that the **syscompdicts_full** table and the **syscompdicts** view display information about all compression dictionaries, not just the active dictionaries.

Syntax:

```
onstat -g ppd [partition number]
                0
```

If you specify a partition number, **onstat -g ppd** prints the partition profile for that partition. If you specify 0, this option prints profiles for all partitions.

Example output

Partition Compression Dictionary Info								
partnum	Version	DbsNum	CrTS	CrLogID	CrLogPos	DrTS	DrLogID	DrLogPos
0x200002	1	2	1229018150	3	577560	0	0	0
0x200003	1	2	1229018150	3	606232	0	0	0
0x300002	1	3	1229018150	3	630808	0	0	0
0x400002	1	4	1229018150	3	655384	0	0	0
0x500002	1	5	1229018150	3	679960	0	0	0

Figure 20-80. **onstat -g ppd** Output

Output description

partnum

Partition number to which the compression dictionary applies

Version

Version of the code that is creating the compression dictionary

DbsNum

Number of the dbspace that the dictionary resides in

CrTS Timestamp that shows when the dictionary was created

CrLogID

Unique ID for the logical log that was created when the dictionary was created

CrLogPos

Position within the logical log when the dictionary was created

DrTS Timestamp that shows when the dictionary was purged

DrLogID

Unique ID for the logical log that was created when the dictionary was purged

DrLogPos

Position within the logical log when the dictionary was purged

onstat -g ppf command: Print partition profiles

Use the **onstat -g ppf *partition_number*** command to display the partition profile for the specified partition number.

Use the **onstat -g ppf** or the **onstat -g ppf 0** command to display the profiles for all partitions. If the TBLSPACE_STATS configuration parameter is set to 0, then the **onstat -g ppf** command displays: Partition profiles disabled.

For more information on the **onstat -g ppf** command, see the *IBM Informix Performance Guide*.

Syntax:

```

>> onstat -g ppf partition_number
0

```

Example output

Partition profiles												
partnum	lkrqs	lkwts	dlks	touts	isrd	iswrt	isrwt	isdel	bfrd	bfwrt	seqsc	rhitratio
0x100001	0	0	0	0	0	0	0	0	0	0	0	0
0x100002	1506	0	0	0	416	4	0	4	1282	20	0	97
0x100003	15	0	0	0	5	0	0	0	20	0	0	75
0x1000a5	0	0	0	0	0	0	0	0	12	0	0	67
0x1000e3	4	0	0	0	1	0	0	0	4	0	0	25
0x200001	0	0	0	0	0	0	0	0	0	0	0	0
0x300001	0	0	0	0	0	0	0	0	0	0	0	0
0x400001	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-81. **onstat -g ppf** command output

Output description

partnum (hex)

The partition number

lkrqs (decimal)

The number of lock requests for a partition

lkwts (decimal)

The number of lock waits for a partition

dlks (decimal)

The number of deadlocks for a partition

touts(decimal)

The number of remote deadlock timeouts for a partition

isrd (decimal)

The number of read operations for a partition

iswrt (decimal)

The number of write operations for a partition

isrwt (decimal)

The number of rewrite or update operations for a partition

isdel (decimal)

The number of delete operations for a partition

bfrd (decimal)

The number of buffer read operations, in pages

bfwrt (decimal)

The number of buffer write operations, in pages

seqsc (decimal)

The number of sequential scans for a partition

rhitratio (percentage)

The ratio of disk read operations to buffer read operations

onstat -g pqs command: Print operators for all SQL queries

Use the **onstat -g pqs** command to display information about the operators used in all of the SQL queries that are currently running.

You can use this command to troubleshoot an application, to find which operators are running for the query and for how long, and how many rows each operator returns. While the EXPLAIN file contains information that will give you a general sense of the query plan, the **onstat -g pqs** command displays the runtime operator information for the query and the query plan.

Syntax:

```

>> onstat -g pqs [sessionid]

```

You can specify one of the following invocations:

Table 20-25. Descriptions of each **onstat -g pqs** command invocation

Invocation	Explanation
onstat -g pqs	Displays a one-line summary for each session.
onstat -g pqs sessionid	Displays information for the session that you specify.

Example output

The following example shows the results when three separate SQL statements are run in different sessions. The statements are:

```

select * from syscolumns;
select * from systables a, systables b;
update t1 set rowsize = rowsize +100;

```

```

Query Operators:
addr      ses-id  opname  phase  rows  time          in1      in2      stmt-type
ae50b3a   23      scan   open   0      00:00.00      0        0      SELECT
af269d0   5        nljoin next   224717 00:01.82  af26a90  aeb4478  SELECT
af26a90   5        scan   next   472    00:00.20      0        0      SELECT
aeb4478   5        scan   next   50     00:01.63      0        0      SELECT
ad3c530   26      scan   open   0      00:00.00      0        0      UPDATE (all)

```

Figure 20-82. **onstat -g pqs** command output

Output description

addr The address of the operator in memory. You can use this address to track which SCAN operator belongs to each JOIN operator.

ses-id The session ID in which the SQL statement was run.

opname
The name of the operator.

phase The phase in which the operator was used. For example OPEN, NEXT, CLOSE.

rows The number of rows that are processed by the operator.

time The amount of time to process the operator. The time is displayed to the millisecond. A time of 01:20.10 is 1 minute, 20 seconds, and 10 milliseconds.

in1 The first (outer) operator in the join.

in2 The second (inner) operator in the join.

stmt-type
The type of SQL statement, such as SELECT, UPDATE, DELETE.

onstat -g prc command: Print sessions using UDR or SPL routine

Use the **onstat -g prc** command to display the number of sessions that are currently using the UDR or SPL routine.

Syntax:

```

▶▶ onstat -g prc ◀◀

```

Example output

```

Stored Procedure Cache:
  Number of lists      : 31
  PC_POOLSIZ          : 50
  Number of entries    : 0
  Number of inuse entries : 0
Stored Procedure Cache Entries:
list# id ref_cnt dropped? heap_ptr procedure name
-----
Stored Procedure Cache is empty.

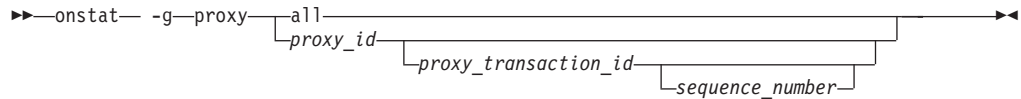
```

Figure 20-83. **onstat -g prc** command output

onstat -g proxy command: Print proxy distributor information

Use the **onstat -g proxy** command to display information about proxy distributors. The output of the **onstat -g proxy** command differs slightly depending on whether the command is run on a primary server or on a secondary server.

Syntax:



Invocation	Explanation
onstat -g proxy	Displays proxy distributor information
onstat -g proxy all	When run on the primary server, displays information about proxy distributors and proxy agent threads. When run on the secondary server, displays information about all sessions currently performing updates to secondary servers.
onstat -g proxy proxy_id proxy_transaction_id sequence_number	This option is valid only on secondary servers. Displays detailed information about the current work being performed by a given proxy distributor. The <i>proxy_transaction_id</i> and <i>sequence_number</i> are optional parameters. When supplied, the first number is considered the <i>proxy_transaction_id</i> , and the second is interpreted as the <i>sequence_number</i> . If the supplied <i>proxy_transaction_id</i> or <i>sequence_number</i> do not exist, the command output is the same as the output for onstat -

Example output using the onstat -g proxy command on a primary server

Secondary Node	Proxy ID	Reference Count	Transaction Count	Hot Row Total
nagpur_sdc1	2619	0	2	0
nagpur_c2	2632	0	1	0
nagpur_sec	2633	0	1	0 I

Figure 20-84. **onstat -g proxy** command output (run from primary server)

Output description

Secondary Node

Name of the secondary server as it is known by the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor.

Example output using the onstat -g proxy command on a secondary server

Primary Node	Proxy ID	Reference Count	Transaction Count	Hot Row Total
nagpur	2619	5	2	0

Figure 20-85. onstat -g proxy command output (run from secondary server)

Output description

Primary Node

Name of the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor.

Example output using the onstat -g proxy all command on a primary server

Secondary Node	Proxy ID	Reference Count	Transaction Count	Hot Row Total
nagpur_sdc1	2619	0	2	0
nagpur_c2	2632	0	1	0
nagpur_sec	2633	0	1	0

TID	Flags	Proxy ID	Source SessID	Proxy TxnID	Current Seq	sqlerrno	iserrno
94	0x00000224	2619	21	1	29	0	0
95	0x00000224	2619	22	2	68	0	0
93	0x00000224	2632	21	2	2	0	0
91	0x00000224	2633	25	1	6	0	0

Figure 20-86. onstat -g proxy all command output (run from primary server)

Output description

Secondary Node

Name of the secondary server as it is known by the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor.

TID

ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the session on the secondary server.

Flags

Flags of the proxy agent thread.

Proxy ID

The ID of the proxy distributor on behalf of which the proxy agent thread (TID) is running.

Source SessID

The ID of the user's session on the secondary server.

Proxy TxnID

The number of the current transaction. These numbers are unique to the proxy distributor.

Current Seq

The sequence number of the current operation in the current transaction.

sqlerrno

The error number of any SQL error (or 0 if no errors).

iserrno

The error number of any ISAM or RSAM error (or 0 if no errors).

Example output using the onstat -g proxy all command on a secondary server

Primary Node		Proxy ID	Reference Count	Transaction Count	Hot Row Total		
nagpur		2619	5	2	0		
Session	Session Ref	Proxy Proxy_id	Proxy TID	Proxy TxnID	Current Seq	Pending Ops	Reference Count
21	2	2619	94	1	29	1	1
22	2	2619	95	2	68	1	1

Figure 20-87. onstat -g proxy all command output (run from secondary server)

Output description

Primary Node

Name of the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor. A hot row is a row on a secondary server that is updated multiple times by more than one client. When a row is updated multiple times, the secondary server reads the before image from the primary server by placing an update lock on the row if the most recent update operation from a different session is not replayed on the secondary server.

Session The session ID

Proxy ID

The ID of the proxy distributor on behalf of which the proxy agent thread (TID) is running.

Proxy TID

Transaction ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the secondary server session.

Proxy TxnID

The number of the current transaction. These numbers are unique to the proxy distributor.

Current Seq

The sequence number of the current operation in the current transaction.

Pending Ops

The number of operations buffered on the secondary server that have not yet been sent to the primary server.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Example output using the *proxy_id* option on a secondary server

This command returns information only on a secondary server.

Proxy TxnID	Reference Count	Pending Ops	ProxySID
1	1	1	3
2	1	1	4

Figure 20-88. **onstat -g proxy proxy_id** command output (run from secondary server)

Output description

Proxy TxnID

The number of the current transaction. These numbers are unique to the proxy distributor.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Pending Ops

The number of operations buffered on the secondary server that have not yet been sent to the primary server.

Proxy SID

Proxy session ID.

Example output using the *proxy_id proxy_transaction_id* options on a secondary server

This command returns information only on a secondary server.

Sequence Number	Operation Type	rowid	Table Name	sqlerrno
28	*Update	526	stores_demo:nilesho.customer	0

Figure 20-89. **onstat -g proxy_id proxy_transaction_id** command output (run from secondary server)

Output description

Sequence Number

The number of the operation.

Operation Type

The type of operation to be performed. One of: Insert, Update, Delete, Other.

rowid The row ID of the row in which to apply the operation.

Table Name

The full table name, trimmed to fit a reasonable length. Format: database.owner.tablename

sqlerrno

The error number of any SQL error (or 0 if no errors).

Example output using the *proxy_id proxy_transaction_id sequence_number* options on a secondary server

This command returns information only on a secondary server.

The output fields are the same as the output fields displayed for the **onstat -g proxy_id proxy_transaction_id** command. While the **onstat -g proxy_id proxy_transaction_id** command displays details for a transaction, the **onstat -g proxy_id proxy_transaction_id sequence_number** displays details for all transaction operations.

```

s
Proxy    Reference Pending ProxySID
TxnID    Count    Ops
61        0        3        22

onstat -g proxy 2788 61

Sequence Operation rowid    Table
Number   Type      Name
960      Update   264    stores_demo:nileshe.customer
961      Update   265    stores_demo:nileshe.orders
962      Update   266    stores_demo:nileshe.items
                                sqlerrno
                                0
                                0
                                0

onstat -g proxy 2788 61 962

Sequence Operation rowid    Table
Number   Type      Name
962      Update   266    stores_demo:nileshe.items
                                sqlerrno
                                0

```

Figure 20-90. **onstat -g proxy_id proxy_transaction_id sequence_number** command output (run from secondary server)

onstat -g que command: Prints ER queue statistics

Use the **onstat -g que** command to display statistics that are common to all queues in Enterprise Replication.

The queuer manages the logical aspects of the queue. The RQM (reliable queue manager) manages the physical queue.

The **onstat -g que** command is used primarily as a debugging tool and most often by Technical Support.

Syntax:

```

▶▶ onstat -g que ◀◀

```

Example output

In the following example, **Element high water mark** shows the maximum size of the transaction buffer header data (metadata) allowed in memory, shown in kilobytes. **Data high water mark** shows the maximum size of transactions for user data allowed in memory, shown in kilobytes.

```

CDR Queuer Statistics:
  Queuer state      : 2
  Local server     : 100
  Element high water mark : 131072
  Data high water mark : 131072
  # of times txns split : 0
  Total # of split txns : 0
  allowed log delta : 30
  maximum delta detected : 4
  Control Key      : 0/00000007
  Synchronization Key : 0/00000003
Replay Table:
  Replay Posn (Disk value): 12/00000018 (12/00000018)
  Replay save interval   : 10
  Replay updates        : 10
  Replay # saves        : 17
  Replay last save time  : (1095118157) 2004/09/13 18:29:17
Send Handles
  Server ID          : 200
  Send state,count   : 0,0
  RQM hdl for trg_send: Traverse handle (0xaf8e018) for thread CDRACK_0 at Head_of_Q,
    Flags: None
  RQM hdl for control_send: Traverse handle (0xaf74018)
    for thread CDRACK_0 at Head_of_Q,  Flags: None
  RQM hdl for sync_send: Traverse handle (0xad6018) for thread CDRACK_0 at Head_of_Q,
    Flags: None
  Server ID          : 200
  Send state,count   : 0,0
  RQM hdl for trg_send: Traverse handle (0xac8b018) for thread CDRACK_1 at Head_of_Q,
    Flags: None
  RQM hdl for control_send: Traverse handle (0xb1ce018) for thread CDRACK_1 at Head_of_Q,
    Flags: None
  RQM hdl for sync_send: Traverse handle (0xadc5018) for thread CDRACK_1 at Head_of_Q,
    Flags: None
  Server ID          : 200
  Send state,count   : 0,0
  RQM hdl for trg_send: Traverse handle (0xae71d8) for thread CDRNsA200 at Head_of_Q,
    Flags: None
  RQM hdl for ack_send: Traverse handle (0xae8c1d8) for thread CDRNsA200 at Head_of_Q,
    Flags: None
  RQM hdl for control_send: Traverse handle (0xae9e1d8) for thread CDRNsA200 at Head_of_Q,
    Flags: None

```

Figure 20-91. **onstat -g que** command output

onstat -g qst command: Print wait options for mutex and condition queues

Use the **onstat -g qst** command to display the wait statistics for mutex queues and condition queues (queues of waiters for a mutex or a condition).

The QSTATS configuration parameter must be set to 1 to enable the collection of statistics. For more information, see “QSTATS Configuration Parameter” on page 1-112.

Syntax:

```

▶▶ onstat — -g qst —▶▶

```

Example output

```
Mutex Queue Statistics
name      nwaits   avg_time max_time avgq maxq nservs   avg_time
ddh chai 1      1354863  1354863  1    1    56      1690

Condition Queue Statistics
name      nwaits   avg_time max_time avgq maxq nservs   avg_time
arrived 1      110008   110008  1    1    0        0
logbf0 21      642      4431   1    2    0        0
logbf1 15      475      2519   1    2    0        0
logbf2 19      596      3274   1    2    0        0
bp_cond 1        0        0      1    1    0        0
```

Figure 20-92. **onstat -g qst** command output

Output description

name (string)

Name of the mutex or condition resource being waited for

nwaits (decimal)

Number of times this resource was waited for

avg_time (decimal)

Average time spent waiting (in microseconds)

max_time (decimal)

Maximum time spent waiting (in microseconds)

avgq (decimal)

Average length of the queue

maxq (decimal)

Maximum length of the queue

nservs (decimal)

Number of times this resource was acquired

avg_time (decimal, microsecond)

Average time the resource was held per acquisition (in microseconds)

onstat -g rah command: Print read-ahead request statistics

Use the **onstat -g rah** command to display information about read-ahead requests.

Syntax:

►►—onstat— -g—rah—►►

Example output

Read Ahead										
# threads	4									
# Requests	150501									
# Continued	1017									
# Memory Failures	0									
Q depth	0									
Last Thread Add	04/06/2011.14:34									
Partition ReadAhead Statistics										
	Buffer	Disk	Hit	Data		Index		Idx/Dat		
Partnum	Reads	Reads	Ratio	# Reqs	Eff	# Reqs	Eff	# Reqs	Eff	
0x200003	2412176	587381	75	0	0	150499	67	0	0	

Figure 20-93. **onstat -g rah** command output

Output description

threads

Number of read-ahead threads

Requests

Number of read-ahead requests

Continued

Number of times a read-ahead request continued to occur

Memory Failures

Number of failed requests because of insufficient memory

Q Depth

Depth of request queue

Last Thread Add

Date and time when the last read-ahead thread was added

Partnum

Partition number

Buffer reads

Number of bufferpool and disk pages that were read

Disk Reads

Number of pages that were read from disk

Hit Ratio

Cache hit ratio for the partition

Reqs

Number of data page read ahead requests. (There are three instances of this output field: for data, the index, and index data.)

Eff

Efficiency of the read-ahead requests. This is the ratio between the number of pages requested by read-ahead operations to the number of pages that were already cached and for which a read-ahead operations was not needed. Values are between 0 and 100. A higher number means that read ahead is beneficial. (There are three instances of this output field: for data, for the index, and for index data.)

onstat -g rbm command: Print a block map of shared memory

Use the **onstat -g rbm** command to display a hexadecimal bitmap of the free and used blocks within the resident segment of shared memory.

Syntax:

►► onstat — -g —rbm —————►►

Example output

```
Block bitmap for resident segment address 0x44000000:
address = 0x440003bc, size(bits) = 3035
used = 3031, largest_free = 4

  0:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
256:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
512:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
768:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
1024:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
1280:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
1536:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
1792:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
2048:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
2304:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
2560:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffffff
2816:ffffffff ffffffffff ffffffffff ffffffffff ffffffffff ffffffff ffffffe00
```

Figure 20-94. onstat -g rbm command output

Output description

Header

- address (hex)**
In-memory starting address of the used/free blocks in the segment
- size (bits)**
Number of bits in the block bitmap; each bit represents one block
- used (blocks)**
Used blocks in the bitmap
- largest_free (blocks)**
Largest run of free blocks

Data

- Bit number (decimal): data (hex)**
Bit number followed by 32 bytes of data (hex)

onstat -g rcv command: Print ER receive manager statistics

Use the **onstat -g rcv** command to display statistics for transactions of the Receive Manager in Enterprise Replication.

The Receive Manager is a set of service routines that pass information between the receive queues and data sync threads.

The **onstat -g rcv** command is used primarily as a debugging tool and by Technical Support. If you suspect that acknowledgment messages are not being applied, you can use this option to check.

Syntax:



The **onstat -g rcv** command has the following formats:

```
onstat -g rcv
onstat -g rcv server_id
onstat -g rcv full
```

The *server_id* modifier causes the command to print only those output messages received from the replication server whose group ID is *server_id*. The **full** modifier causes the command to print all statistics.

Example output

The following example shows output for the **onstat -g rcv full** command.

```

Receive Manager global  block 0D452018
  cdrRM_inst_ct:                5
  cdrRM_State:                   00000000
  cdrRM_numSleepers:             3
  cdrRM_DsCreated:               3
  cdrRM_MinDSThreads:           1
  cdrRM_MaxDSThreads:           4
  cdrRM_DSBlock                  0
  cdrRM_DSParallelPL            0
  cdrRM_DSFailRate               0.000000
  cdrRM_DSNumRun:                35
  cdrRM_DSNumLockTimeout         0
  cdrRM_DSNumLockRB              0
  cdrRM_DSNumDeadLocks           0
  cdrRM_DSNumPCommits            0
  cdrRM_ACKwaiting               0
  cdrRM_totSleep:                77
  cdrRM_Sleeptime:               153
  cdrRM_Workload:                0
  cdrRM_optscale:                4
  cdrRM_MinFloatThreads:         2
  cdrRM_MaxFloatThreads:         7
  cdrRM_AckThreadCount:          2
  cdrRM_AckWaiters:              2
  cdrRM_AckCreateStamp: Wed Sep 08 11:47:49 2004
  cdrRM_DSCreateStamp: Wed Sep 08 14:16:35 2004
  cdrRM_acksInList:              0
  cdrRM_BlobErrorBufs:           0

Receive Parallelism Statistics
Srvr Tot.Txn. Pending Active MaxPnd MaxAct AvgPnd AvgAct CommitRt
  1    35      0      0    21    3    7.00    1.63    0.00
  5     3      0      0     1     1    1.00    1.00    0.02
  6     6      0      0     1     1    1.00    1.00    0.21
Tot Pending:0 Tot Active:0 Avg Pending:5.77 Avg Active:1.50
Commit Rate:0.01

Time Spent In RM Parallel Pipeline Levels
Lev. TimeInSec  Pcnt.
  0    17405 100.00%
  1      0  0.00%
  2      0  0.00%

Statistics by Source
Server 1
Repl Txn Ins Del Upd Last Target Apply Last Source Commit
65541 23  0  1 616 2004/09/08 14:20:15 2004/09/08 14:20:15
65542 11  0  0 253 2004/09/08 14:19:33 2004/09/08 14:19:33
65545 1  0 67  0 2004/09/08 14:20:37 2004/09/08 14:20:37
Server 5
Repl Txn Ins Del Upd Last Target Apply Last Source Commit
65541 3  0  0 81 2004/09/08 16:36:10 2004/09/08 16:36:09
Server 6
Repl Txn Ins Del Upd Last Target Apply Last Source Commit
65548 6  0  0 42 2004/09/08 16:37:59 2004/09/08 16:37:58

```

Figure 20-95. **onstat -g rcv full** command output

Fields in the Receive Manager global section

The **onstat -g rcv** command includes the Receive Manager global section. In this section, the following fields have the meanings shown:

Field	Description
cdrRM_DSParallelPL	Shows the current level of Apply Parallelism, 0 (zero) being the highest
cdrRM_DSNumLockTimeout cdrRM_DSNumLockRB cdrRM_DSNumDeadLocks	Indicate the number of collisions between various apply threads
cdrRM_acksinList	Shows acknowledgments that have been received but not yet processed

Fields in the Receive Parallelism Statistics section

The **onstat -g rcv** command includes the Receive Parallelism Statistics section, a summary of the data sync threads by source server.

Field	Description
Server	Source server ID
Tot.Txn.	Total number of transactions applied from this source server
Pending	Number of current transactions in the pending list for this source server
Active	Number of current transactions currently being applied from this source server
MaxPnd	Maximum number of transactions in the pending list queue
MaxAct	Maximum number of transaction in the active list queue
AvgPnd	Average depth of the pending list queue
AvgAct	Average depth of the active list queue
CommitRt	Commit rate of transaction from this source server based on transactions per second

Output in the Statistics by Source section

The Statistics by Source section of the **onstat -g rcv** command shows the following information for each source server. For each replicate ID:

- The number of transactions applied from the source servers
- The number of inserts, deletes, and updates within the applied transactions
- The timestamp of the most recently applied transaction on the target server
- The timestamp of the commit on the source server for the most recently applied transaction

onstat -g rea command: Print ready threads

Use the **onstat -g rea** command to display information about the virtual processor threads whose current status is ready.

Syntax:

```
►► onstat -g rea ◀◀
```

Example output

```
Ready threads:
tid      tcb      rstcb  prty   status  vp-class  name
6        536a38  406464  4      ready   3cpu     main_loop()
28       60cfe8  40a124  4      ready   1cpu     onmode_mon
33       672a20  409dc4  2      ready   3cpu     sqlexec
```

Figure 20-96. **onstat -g rea** command output

onstat -g rep command: Print ER schedule manager events

Use the **onstat -g rep** command to display information about events that are in the queue for the Schedule Manager for Enterprise Replication.

The **onstat -g rep** command is used primarily as a debugging tool and most often by Technical Support.

Syntax:

```
onstat -g rep [replname]
```

The **onstat -g rep** command has the following formats:

```
onstat -g rep
onstat -g rep replname
```

The *repl_name* modifier limits the output to those events originated by the replicate named *repl_name*.

Example output

The following example shows sample output for the **onstat -g rep** command.

```
Schedule manager Cb: add7e18 State: 0x8100 <CDRINIT,CDRRUNNING>
Event      Thread      When
=====
CDRDS      CDREvent      00:00:20
```

Figure 20-97. **onstat -g rep** command output

onstat -g rqm command: Print low-level queue statistics

Use the **onstat -g rqm** commands to display statistics and the contents of the low-level queues (send queue, receive queue, ack send queue, sync send queue, and control send queue) that are managed by the Reliable Queue Manager (RQM) in Enterprise Replication.

The RQM manages the insertion and removal of items to and from the various queues. The RQM also manages spooling of the in-memory portions of the queue to and from disk. The **onstat -g rqm** command displays the contents of the queue, size of the transactions in the queue, how much of the queue is in memory and on disk, the location of various handles to the queue, and the contents of the various progress tables. You can choose to print information for all queues or for just one queue by using one of the modifiers described below.

If a queue is empty, no information is printed for that queue.

Syntax:

```
►► onstat -g rqm modifier ►►
```

The **onstat -g rqm** command has the following formats:

```
onstat -g rqm
onstat -g rqm modifier
```

The following table describes the values for *modifier*.

Modifier	Action
ACKQ	Prints the ack send queue
BRIEF	Prints a brief summary of the number of transactions in each of the queues and the replication servers for which the data is queued. Use this modifier to quickly identify sites where a problem exists. If large amounts of data are queued for a single server, then that server is probably down or off the network.
CNTRLQ	Prints the control send queue
FULL	Prints full information about every in-memory transaction for every queue
RECVQ	Prints the receive queue
SBSPACES	Prints information about the sbspaces configured for ER
SENDQ	Prints the send queue
SYNCQ	Prints the sync send queue
VERBOSE	Prints all the buffer headers in memory

When you specify a modifier to select a specific queue, the command prints all the statistics for that queue and information about the first and last in-memory transactions for that queue.

The other modifiers of the **onstat -g rqm** command are used primarily as a debugging tool and most often by Technical Support.

Output sections for the SENDQ modifier

The output for the SENDQ modifier contains the following sections:

- RQM Statistics for Queue—a summary of current and historical information for the queue. This includes the number of transactions in the queue, how many are spooled, how many bytes they are using, some maximum statistics, and the high water marks that will trigger stably storing transactions in the **syscdr** tables.
- First Txn—information about the first transaction in the queue. To check if the queue is draining, you can run the **onstat -g rqm** command several times and see if the first transaction's RQM key is changing. The RQM key has the following format: *Server_ID/Commit_unique_logID/Commit_log_position/Sequence*. If it is not draining, the target server may be offline or some other problem is occurring. The NeedAck field shows from which server the transaction is waiting for an acknowledgment. You can use this bitmap mask with the output from the **onstat -g cat** command to determine the name of the server which server Enterprise Replication is waiting on for an acknowledgment.

- Last Txn—information about the last transaction in the queue
- Traverse handle—lists the handles used for threads
- Progress table—provides information about the progress of each replicate under the headers: Server, Group, Bytes Queued, Acked, and Sent. The Group field shows the replicate ID. The Acked field shows what has been acknowledged. The Sent field shows which entries are now in transit. Both the Acked and the Sent field show the RQM key, which has the following format:
Server_ID/Commit_unique_logID/Commit_log_position/Sequence.

Example output

The following example shows output for the **onstat -g rqm SENDQ** command.

```
RQM Statistics for Queue (0x0D3DF018) trg_send
Transaction Spool Name: trg_send_stxn
Insert Stamp: 35/0
Flags: SEND_Q, SPOOLED, PROGRESS_TABLE, NEED_ACK
Txns in queue:          35
Log Events in queue:    0
Txns in memory:         35
Txns in spool only:     0
Txns spooled:           0
Unspooled bytes:        176206
Size of Data in queue:  176206 Bytes
Real memory in use:     176206 Bytes
Pending Txn Buffers:    0
Pending Txn Data:       0 Bytes
Max Real memory data used: 176206 (2457600) Bytes
Max Real memory hdrs used 65988 (2457600) Bytes
Total data queued:      176206 Bytes
Total Txns queued:      35
Total Txns spooled:     0
Total Txns restored:    0
Total Txns recovered:   0
Spool Rows read:        0
Total Txns deleted:     0
Total Txns duplicated:  0
Total Txn Lookups:      363

First Txn (0x0D60C018) Key: 1/9/0x000d4bb0/0x00000000
Txn Stamp: 1/0, Reference Count: 0.
Txn Flags: Notify
Txn Commit Time: (1094670993) 2004/09/08 14:16:33
Txn Size in Queue: 5908
First Buf's (0x0D31C9E8) Queue Flags: Resident
First Buf's Buffer Flags: TRG, Stream
NeedAck: Waiting for Acks from <[0004]>
No open handles on txn.

Last Txn (0x0D93A098) Key: 1/9/0x00138ad8/0x00000000
Txn Stamp: 35/0, Reference Count: 0.
Txn Flags: Notify
Txn Commit Time: (1094671237) 2004/09/08 14:20:37
Txn Size in Queue: 6298
First Buf's (0x0D92FFA0) Queue Flags: Resident
First Buf's Buffer Flags: TRG, Stream
NeedAck: Waiting for Acks from <[0004]>

Traverse handle (0x0D045018) for thread CDRNsA3 at txn (0x0D93A098)
End_of_Q,Flags: None
Traverse handle (0x0D08E018) for thread CDRNsA4 at txn (0x0D93A098)
End_of_Q,Flags: None
Traverse handle (0x0D523018) for thread CDRNsA5 at txn (0x0D93A098)
End_of_Q,Flags: None
Traverse handle (0x0D0D9018) for thread CDRNsA6 at txn (0x0D93A098)
End_of_Q,Flags: None
```

```

    Traverse handle (0x0D4041D8) for thread CDRNsA2 at Head_of_Q,Flags: None
    Traverse handle (0x0D3F01D8) for thread CDRNrA2 at Head_of_Q, Flags: None
    Traverse handle (0x0D045018) for thread CDRNsA3 at txn (0x0D93A098)
End_of_Q,Flags: None
    Traverse handle (0x0D31C018) for thread CDRNrA3 at Head_of_Q, Flags: None
    Traverse handle (0x0D08E018) for thread CDRNsA4 at txn (0x0D93A098)
End_of_Q,Flags: None
    Traverse handle (0x0D4C8018) for thread CDRNrA4 at Head_of_Q, Flags: None
    Traverse handle (0x0D523018) for thread CDRNsA5 at txn (0x0D93A098)
End_of_Q,Flags: None
    Traverse handle (0x0D57F018) for thread CDRNrA5 at Head_of_Q, Flags: None
    Traverse handle (0x0D0D9018) for thread CDRNsA6 at txn (0x0D93A098)
End_of_Q,Flags: None

```

Server	Group	Bytes Queued	Acked	Sent
6	0x10009	0	1/9/138ad8/0	- 1/9/138ad8/0
5	0x10009	0	1/9/138ad8/0	- 1/9/138ad8/0
4	0x10009	0	1/9/138ad8/0	- 1/9/138ad8/0
3	0x10009	0	1/9/138ad8/0	- 1/9/138ad8/0
2	0x10009	4154	ffffff/ffffff/ffffff/ffffff	- 1//138ad8/0
6	0x10006	0	1/9/12d8f8/0	- 1//12d8f8/0
5	0x10006	0	1/9/12d8f8/0	- 1//12d8f8/0
4	0x10006	0	1/9/12d8f8/0	- 1/9/12d8f8/0
3	0x10006	0	1/9/12d8f8/0	- 1/9/12d8f8/0
2	0x10006	31625	ffffff/ffffff/ffffff/ffffff	- 1/9/12d8f8/0

onstat -g rss command: Print RS secondary server information

Use the **onstat -g rss** commands to display information about remote standalone secondary servers.

Syntax:

```

>> onstat -g rss
    -verbose
    -log
    -server_name

```

The output of the **onstat -g rss** command differs slightly depending on whether the command is run on the primary server or on the RS secondary server.

Invocation	Explanation
onstat -g rss	Displays brief RS secondary server information
onstat -g rss verbose	Displays detailed RS secondary server information
onstat -g rss log	Displays log information. This command is only applicable when run on the primary server.
onstat -g rss server_name	Displays information about a specific RS secondary server. This command is only applicable when run on the primary server.

Example output (primary server)

```

Local server type: Primary
Index page logging status: Enabled
Index page logging was enabled at: 2009/08/31 09:35:22
Number of RSS servers: 1

RSS Server information:

RSS Server control block: 0x5fdd9740
RSS server name: serv3
RSS server status: Active
RSS connection status: Connected
Log transmission status: Active
Next log page to send(log id,page): 53,117632
Last log page acked(log id,page): 53,115615
Time of Last Acknowledgment: 2009-08-31.14:14:09
Pending Log Pages to be ACKed: 1984
Approximate Log Page Backlog:97104
Sequence number of next buffer to send: 3676
Sequence number of last buffer acked: 3612
Supports Proxy Writes: Y

```

Figure 20-98. **onstat -g rss verbose** command output (run on primary server)

Output description (primary server)

Local server type

Primary or RSS (remote standalone secondary) server type

Index page logging status

Displays whether index page logging is enabled or disabled between primary server and secondary server

Index page logging was enabled at

Date and time that index page logging was enabled

Number of RSS servers

Number of RS secondary servers connected to the primary server

RSS Server control block

RS secondary server control block

RSS Server name

Name of RS secondary server

RSS Server status

Displays whether RS secondary server is active or not

RSS Connection status

Connection status of RS secondary server

Log transmission status

Displays whether log transmission is active or inactive

Next log page to send (log id, page)

The log ID and page number of the next log page that will be sent

Last log page acked (log id, page)

The log ID and page number of the last acknowledged log

Time of Last Acknowledgment

The time at which the last log was acknowledged

Pending Log pages to be ACKed

The number of logs sent but not yet acknowledged

Approximate Log Page Backlog

The difference between the number of logs that were sent and the end of the logical log

Sequence number of next buffer to send

The sequence number of the next buffer to be sent

Sequence number of last buffer acked

The sequence number of the last acknowledged buffer

Supports Proxy Writes

Displays whether the server is currently configured to allow updates to secondary servers. **Y** = supports updates to secondary servers, **N** = does not support updates to secondary servers.

Example output (RS secondary server)

```
Local server type: RSS
Server Status : Active
Source server name: cdr_ol_nag_1
Connection status: Connected
Last log page received(log id,page): 7,877
```

Figure 20-99. **onstat -g rss** command output (run on RS secondary server)

Output description (RS secondary server)

Local server type

Primary or RSS (remote standalone secondary) server type

Server Status

Displays whether RS secondary server is active

Source server name

Name of the primary server

Connection status

Connection status of RS secondary server

Last log page received (log id,page)

Most recent log ID and page received

Example output with log option (primary server)

```
Log Pages Snooped:
RSS Srv      From    From    Tossed
name         Cache   Disk    (LBC full)
cdr_ol_nag_1_c1 1368    1331    0
cdr_ol_nag_1_c2 1357    1342    0
cdr_ol_nag_1_c3 1356    1343    0
```

Figure 20-100. **onstat -g rss log** command output (run on primary server)

Output description with log option (primary server)

Log Pages Snooped

Statistics for each RS secondary server

RSS Srv name
 RS secondary server name

From Cache
 From cache number

From Disk
 Log from disk

Tossed (LBC full)
 Number of log pages discarded as a result of the LBC becoming full

onstat -g rwm command: Print read and write mutexes

Use the **onstat -g rwm** command to display information about read, write, and waiting mutex threads, and to list the addresses of the tickets that these threads have acquired.

Syntax:

►► onstat — -g—rwm ————— ◀◀

Example output

```
MUTEX  NAME  write/read/wait  tcb list
<address> <name>  first mutex
  Writer  ticket = <ticket address>  tcb=<thread address> <thread name>
  Readers ticket = <ticket address>  tcb=<thread address> <thread name>
  Waiters ticket = <ticket address>  tcb=<thread address> <thread name>
<address> <name>  second mutex
  Writer  ticket = <ticket address>  tcb=<thread address> <thread name>
  Readers ticket = <ticket address>  tcb=<thread address> <thread name>
  Waiters ticket = <ticket address>  tcb=<thread address> <thread name>
....
....
....
<address> <name>  last mutex
  Writer  ticket = <ticket address>  tcb=<thread address> <thread name>
  Readers ticket = <ticket address>  tcb=<thread address> <thread name>
  Waiters ticket = <ticket address>  tcb=<thread address> <thread name>
```

Figure 20-101. **onstat -g rwm** command output

Output description

tcb List of thread addresses

Writer List of write threads

Readers
 List of read threads

Waiters
 List of waiting threads

ticket Address of ticket acquired by the thread

onstat -g sch command: Print VP information

Use the **onstat -g sch** command to display information about thread migration and the number of semaphore operations, spins, and busy waits for each virtual processor.

Syntax:

►► onstat — -g — sch ————— ◀◀

Example Output

VP Scheduler Statistics:					
vp	pid	class	semops	busy waits	spins/wait
1	3284	cpu	23997	0	0
2	1340	adm	0	0	0
3	4624	lio	2	0	0
4	3320	pio	2	0	0
5	6076	aio	7710	0	0
6	4580	msc	46	0	0
7	3428	soc	7	0	0
8	2308	soc	1	0	0

Thread Migration Statistics:								
vp	pid	class	steal-at	steal-sc	idlv-at	idlv-sc	inl-polls	Q-ln
1	3284	cpu	0	0	0	0	0	0
2	1340	adm	0	0	0	0	0	0
3	4624	lio	0	0	0	0	0	0
4	3320	pio	0	0	0	0	0	0
5	6076	aio	0	0	0	0	0	0
6	4580	msc	0	0	0	0	0	0
7	3428	soc	0	0	0	0	0	0
8	2308	soc	0	0	0	0	0	0

Figure 20-102. **onstat -g sch** command output

onstat -g scn command: Print scan information

Use the **onstat -g scn** command to display the status of a current scan and information on the scan. The **onstat -g scn** command also identifies whether a scan is a light or bufferpool scan.

If you have a long-running scan, you might want to use this command to check the progress of the scan, to determine how long the scan will take before it completes, and to see whether the scan is a light scan or a bufferpool scan.

Syntax:

►► onstat — -g — scn ————— ◀◀

Example Output

```

Light Scan Info
descriptor  address          next_lpage  next_ppage      ppage_left  bufcnt  look_aside

RSAM batch sequential scan info

SesID Thread Partnum Rowid Rows Scan'd Scan Type Lock Mode Notes
48 68 10016e 12bb09 43146 Light Table Look aside,
40 47 100106 101 0 Buffpool +Test Must copy

RSAM batch index scan info

SesID Thread Partnum Scan Type Lock Mode Notes

```

Figure 20-103. **onstat -g scn** output

Output Description

descriptor (decimal)

Light scan ID

address (hex)

Memory address of the light scan descriptor

next_lpage (hex)

Next logical page address to scan

next_ppage (hex)

Next physical page address to scan

ppage_left (decimal)

Number of physical pages left to scan in the current extent

bufcnt Number of light scan buffers used for this light scan

look_aside

Whether look aside is needed for this light scan (Y = yes, N = no). Look asides occur when a thread needs to examine the buffer pool for existing pages to obtain the latest image of a page being light scanned.

SesID Session ID

Thread

Thread ID

Partnum

Partition number

Rowid

Current row ID

Rows Scan'd

Number of rows that have been scanned

Scan Type

Either:

- Bufferpool
- Light (light scan)

.

Lock Mode

The type of acquired lock or no lock:

- Table (table-level lock acquired)
- Slock (share locks acquired)

- Ulock (update locks acquired)
- blank (no locks acquired)

This column can also show one of the following values:

- +Test (The scan tested for a conflict with the specified lock type; the lock was not acquired.)
- +Keep (The acquired locks will be held until end of session instead of the end of the transaction.)

Notes This column can show one of the following values:

- Look aside

The light scan is performing look aside.

The light scan reads blocks of pages directly from disk into large buffers, rather than getting each page from the buffer manager. In some cases, this process requires the light scan to check the buffer pool for the presence of each data page that it processes from one of its large buffers; this process is called *look aside*. If the page is currently in the buffer pool, the light scan will use that copy instead of the one in the light scan large buffer. If the page is not in the buffer pool, the light scan will use the copy that the light scan read from disk into its large buffer. If the light scan is performing look aside, the performance of the scan is slightly reduced.

In many cases, the light scan can detect that it is impossible for the buffer pool to have a newer version of the page. In these situations, the light scan will not check the buffer pool, and the look aside note will be absent.

- Forward row lookup

The server is performing a light scan on a table that has rows that span pages. The light scan must access and use the buffer pool to get the remainder pieces of any rows that are not completely on the home page.

onstat -g sds command: Print SD secondary server information

Use the **onstat -g sds** command to display information about shared-disk secondary servers.

Syntax:

```

>> onstat -g sds [verbose server_name]

```

The output of the **onstat -g sds** command differs slightly depending on whether the command is issued on the primary server or on the SD secondary server.

Invocation	Explanation
onstat -g sds	Displays brief SD secondary server information
onstat -g sds verbose	Displays detailed SD secondary server information
onstat -g sds server_name	Displays information about a specific SD secondary server. When <i>server_name</i> is specified, the command must be issued from the primary server.

Example output (primary server)

```
Local server type: Primary
Number of SDS servers:1

SDS server information

SDS srv      SDS srv      Connection    Last LPG sent  Supports
name         status       status        (log id,page)  Proxy Writes
C_151162     Active       Connected     554,4998       Y
```

Figure 20-104. **onstat -g sds** command output (run from primary server)

Output description (primary server)

Local server type

Primary or SDS (shared disk secondary) server type

Number of SDS servers

Number of SD secondary servers connected to the primary server

SDS Srv name

Name of SD secondary server

SDS Srv status

Displays whether SD secondary server is active

Connection status

Displays whether SD secondary server is connected

Last LPG sent (log id, page)

Most recent LPG log ID and page

Supports Proxy Writes

Displays whether the server is currently configured to allow updates to secondary servers. **Y** = supports updates to secondary servers, **N** = does not support updates to secondary servers.

Example output with verbose option (primary server)

```

Number of SDS servers:1
Updater node alias name :serv1

SDS server control block: 0x5fd51510
server name: serv2
server type: SDS
server status: Active
connection status: Connected
Last log page sent(log id,page):54,84732
Last log page flushed(log id,page):54,84732
Last log page acked (log id, page):54,84732
Last LSN acked (log id,pos):54,347062296
Approximate Log Page Backlog:0
Current SDS Cycle:2
Aked SDS Cycle:2
Sequence number of next buffer to send: 10
Sequence number of last buffer acked: 9
Time of lask ack:2009/08/31 14:16:11
Supports Proxy Writes: Y

```

Figure 20-105. **onstat -g sds verbose** command output (run from primary server)

Output description with verbose option (primary server)

Number of SDS servers

Number of attached SDS (shared disk secondary) servers

Updater node alias name

Name of primary server

SDS server control block

SD secondary server control block

server name

The name of the server

server type

The type of server

server status

Displays whether the server is active or inactive

connection status

Status of connection between primary and secondary server

Last log page sent (log id, page)

Log ID and page of most recent log page sent

Last log page flushed (log id, page)

Log ID and page of the most recent log page flushed

Last log page acked (log id, pos)

Most recent log page acknowledged

Last LSN acked (log id, pos)

Most recent log sequence number that was acknowledged

Approximate Log Page Backlog

The number of logs waiting to be sent

Current SDS Cycle

Used internally by IBM support to monitor coordination of the primary server with the SDS server

Acked SDS Cycle

Used internally by IBM support to monitor coordination of the primary server with the SDS server

Sequence number of next buffer to send

Sequence number of next buffer to send

Sequence number of last buffer acked

Sequence number of next buffer acknowledged

Time of last ack

Date and time of last log acknowledgment

Supports Proxy Writes

Displays whether the server is currently configured to allow updates to secondary servers. **Y** = supports updates to secondary servers, **N** = does not support updates to secondary servers.

Example output with verbose option (SD secondary server)

```
SDS server control block: 0xb299880
Local server type: SDS
Server Status : Active
Source server name: cdr_ol_nag_1
Connection status: Connected
Last log page received(log id,page): 7,884
Next log page to read(log id,page):7,885
Last LSN acked (log id,pos):7,3621272
Sequence number of last buffer received: 0
Sequence number of last buffer acked: 0
Current paging file:/work1/nagaraju/dbspaces/page_cdr_ol_nag_1_sdc1_
Current paging file size:2048
Old paging file:/work1/nagaraju/dbspaces/page_cdr_ol_nag_1_sdc1_
Old paging file size:10240
```

Figure 20-106. **onstat -g sds verbose** command output (run from SD secondary server)

Output description with verbose option (SD secondary server)

SDS server control block

SD secondary server control block

Local server type

Primary or SDS (shared disk secondary) server type

Server status

Displays whether SD secondary server is active

Source server name

Displays name of primary server

Connection status

Displays whether SD secondary server is connected

Last log page received (log id, page)

Most recent log page received

Next log page to read (log id,page)

Next log page in sequence to read

Last LSN acked (log id,pos)

Most recent LSN acknowledged

Sequence number of last buffer received
 Sequence number of last buffer received

Sequence number of last buffer acked
 Sequence number of last buffer acknowledged

Current paging file
 Name of current paging file

Current paging file size
 Size of current paging file

Old paging file
 Name of previous paging file

Old paging file size
 Size of previous paging file

onstat -g seg command: Print shared memory segment statistics

Use the **onstat -g seg** command to display the statistics for shared memory segments.

This command shows how many segments are attached and their sizes. For information about running the **onstat -g seg** command on a dump file created without the buffer pool, see “Running **onstat** Commands on a Shared Memory Dump File” on page 20-25.

Syntax:

►► onstat — -g — seg —————►►

Example output

id	key	addr	size	ovhd	class	blkused	blkfree
8945678	52604801	44000000	133652480	1006264	R	32627	3
8978447	52604802	4bf76000	131072000	769136	V	17181	14819
Total:	-	-	264724480	-	-	49808	14822

(* segment locked in memory)
 Virtual segment low memory reserve (bytes):4194304
 Low memory reserve used 0 times and used maximum block size 0 bytes

Figure 20-107. **onstat -g seg** command output

Output description

id The ID of the shared memory segment

key The shared memory key associated with the shared memory segment ID

addr The address of the shared memory segment

size The size of the shared memory segment in bytes

ovhd The size of the shared memory segment control information (overhead) in bytes

class The class of the shared memory segment (R is for Resident, V is for Virtual, VX is for Virtual Extended, and M is for Message.)

blkused

The number of blocks of used memory

blkfree

The number of blocks of free memory

Virtual segment low memory reserve (bytes)

The size of reserved memory for use when critical activities are needed and the server has limited free memory, specified in bytes (You specify reserved memory in the LOW_MEMORY_RESERVE configuration parameter.)

Low memory reserve used 0 times and used maximum block size 0 bytes)

The number times that the server has used the reserved memory and the maximum memory needed

Related reference:

“LOW_MEMORY_RESERVE configuration parameter” on page 1-90

onstat -g ses command: Print session-related information

Use the **onstat -g ses** command to display information about the session.

By default, only the DBSA can view **onstat -g ses** syssqltrace information. However, when the UNSECURE_ONSTAT configuration parameter is set to 1, all users can view this information.

Syntax:

►► onstat -g ses sessionid ►►

You can specify one of the following invocations.

Invocation

Explanation

onstat -g ses

Displays a one-line summary for each session

onstat -g ses sessionid

Displays information for a specific session

Example output for all sessions

session					#RSAM	total	used	dynamic
id	user	tty	pid	hostname	threads	memory	memory	explain
24	informix	-	0	-	0	12288	7936	off
23	informix	-	17602	carson	1	57344	48968	off
3	informix	-	0	-	0	12288	9168	off
2	informix	-	0	-	0	12288	7936	off

Figure 20-108. **onstat -g ses** command output

Example output for a specific session

```

session          effective
id      user      user      tty  pid  hostname #RSAM  total  used  dynamic
53      informix -      36   18638 apollo11 1      73728  63048  off

Program :
/usr/informix/bin/dbaccess

tid      name      rstcb      flags      curstk      status
77      sqlxec  4636ba20    Y--P---    4240        cond wait  sm_read  -

Memory pools      count 1
name      class addr      totalsize freesize  #allocfrag #freefrag
53        V      4841d040    73728     10680     84         6

name      free      used      name      free      used
overhead  0      3288     scb       0      144
opentable 0      2904     filetable 0      592
log        0      16536    temprec   0      2208
gentcb     0      1656     ostcb     0      2920
sqscb      0      21296    sql       0      72
hashfiletab 0      552     osenv     0      2848
sqtcb      0      7640     fragman   0      392

sqscb info
scb      sqscb      optofc      pdqpriority optcompind directives
481b70a0 483e2028    0           0           0           1

Sess      SQL      Current      Iso Lock      SQL  ISAM F.E.
Id      Stmt type      Database      Lvl Mode      ERR  ERR  Vers  Explain
53      -      sysmaster      CR  Not Wait    0    0    9.24  Off

Last parsed SQL statement :
Database 'sysmaster@1x1'

Xadatasources participated in this session :
Xadatasource name      RMID      Active
xabasicdb@atmol10:sitaramv.xads_t3_i1      6      YES
xabasicdb@atmol10:sitaramv.xads_t2_i1      4      YES
xabasicdb@atmol10:sitaramv.xads_t1_i3      3      YES
xabasicdb@atmol10:sitaramv.xads_t1_i2      2      YES
xabasicdb@atmol10:sitaramv.xads_t1_i1      1      YES
xabasicdb@atmol10:sitaramv.xads_t2_i2      5      NO

DRDA client info
Userid:
Wrkstnname: nemea
Applname: db2jcc_application
Acctng: JCC03510nemea
Programid:
Autocommit:
Packagepath:

```

Figure 20-109. **onstat -g sesssessionid** command output

Output description: session section

Session id

The session ID

user The username who started the session

tty The tty associated with the front end for this session

<i>pid</i>	The process ID associated with the front end for this session
<i>hostname</i>	The hostname from which this session has connected
<i>#RSAM threads</i>	The number of RSAM thread allocated for this session
<i>total memory</i>	The amount of memory allocated for this session
<i>used memory</i>	The amount of memory actually used by this session
<i>dynamic explain</i>	Generate explain output of the SQL statements of the session (on or off)

Output description: program section

Displays the full path of the client program that is used in your session. Use the client program information to monitor or stop access to the database.

Output description: threads section

Although this section has no title, the following output displays information about threads.

<i>tid</i>	The thread ID
<i>name</i>	The name of the thread
<i>rstcb</i>	RSAM control block
<i>flags</i>	Describes the status of the thread using the following codes:
<i>Position 1</i>	
<i>B</i>	Waiting on a buffer
<i>C</i>	Waiting on a checkpoint
<i>G</i>	Waiting on a logical-log buffer write
<i>L</i>	Waiting on a lock
<i>S</i>	Waiting on a mutex
<i>T</i>	Waiting on a transaction
<i>X</i>	Waiting on a transaction cleanup
<i>Y</i>	Waiting on a condition
<i>Position 2</i>	
<i>*</i>	An asterisk in this position means that the thread encountered an I/O failure in the middle of a transaction
<i>Position 3</i>	
<i>A</i>	Archive thread
<i>B</i>	Begin work
<i>P</i>	Begin Prepare or Prepared work
<i>X</i>	XA prepared
<i>C</i>	Committing or committed

R Aborting or aborted
H Heuristically aborted or heuristically rolling back

Position 4

P Primary thread

Position 5

R Reading

X Critical section

Position 6

R Recovery thread

Position 7

M Monitor thread

D Daemon thread

C Cleaner

F Flusher

B B-tree scanner

curstk Current stack size

status Current thread status

Output description: memory pools header section

The information is repeated for each session pool.

name Name of pool

class Class of the memory where the pool is allocated from. R is for Resident, V is for Virtual, and M is for Message

addr Address of the pool structure

totalsize

Total size of the memory acquired by the pool in bytes

freesize Number of bytes free in the pool

#allocfrag

Number of allocated memory fragments in the pool

#freefrag

Number of free fragments in the pool

Output description: Memory pools section

name Name of a component which has allocated memory from the pool

free Number of bytes freed

used Number of bytes allocated

Output description: sqscb info section

scb The session control block. This is the address of the main session structure in shared memory

sqscb SQL level control block of the session

optofc The current value of the **OPTOFC** environment variable or ONCONFIG configuration file setting

pdqpriority
The current value of the **PDQPRIORITY** environment variable or ONCONFIG configuration file setting

optcompind
The current value of the **OPTCOMPIND** environment variable or ONCONFIG configuration file setting

directives
The current value of the **DIRECTIVES** environment variable or ONCONFIG configuration file setting

Output description: SQL section

Displays SQL information for the specified session. This section contains the same information that is output from the **onstat -g sql** command. See “**onstat -g sql** command: Print SQL-related session information” on page 20-172.

Output description: Last parsed SQL statement section

The Last parsed SQL statement section contains the same information that is output from the **onstat -g sql** command. See “**onstat -g sql** command: Print SQL-related session information” on page 20-172.

Output description: Xadatasources participated in this session section

The Xadatasources participated in this session section shows information about the XA data sources that are available during the session, their resource manager identifiers, and whether they are currently active.

Xdatasource name
The XA data source that participated in the session

RMID The identifier of the resource manager for the corresponding XA data source

Active Whether the XA data source is still active

Output description: DRDA client info section

The **DRDA client info** section shows information about Distributed Relational Database Architecture (DRDA) connections to clients.

Userid
User ID of the client user

Wrkstnname
Name of the client workstation

Applname
Name of the client application, for example db2jcc_application

Acctng
Accounting string from the client, for example JCC03510nemea

Programid
Client program identifier (not used by Informix)

Autocommit

Default transaction autocommit mode for Informix data sources

Packagepath

Client package path (not used by Informix)

onstat -g sle command: Print all sleeping threads

Use the **onstat -g sle** command to print all sleeping threads.

Syntax:

```
➤ onstat -g sle ➤
```

Example output

Current Admin VP sleep period: 10 millisecs				
Sleeping threads with timeouts: 21 threads				
tid	v_proc	rstcb	name	time
49	1	b3b13a8	onmode_mon	0.02
5	1	0	Cosvr Avail Mgr	0.05
42	1	b3ad028	main_loop()	0.08
9	3	b3ad6e8	xtm_svcc	0.64
14	5	0	mgmt_thd_5	0.65
13	4	0	mgmt_thd_4	0.65
4	1	0	mgmt_thd_1	0.65
6	3	0	dfm_svc	0.98
33	13	0	mgmt_thd_13	1.54
27	10	0	mgmt_thd_10	1.54
21	7	0	mgmt_thd_7	1.54
12	3	0	mgmt_thd_3	1.76
29	11	0	mgmt_thd_11	1.76
23	8	0	mgmt_thd_8	2.08
31	12	0	mgmt_thd_12	2.08
35	14	0	mgmt_thd_14	2.98
19	6	0	mgmt_thd_6	3.00
25	9	0	mgmt_thd_9	3.00
37	3	0	sch_rgm	3.48
44	5	b3af8a8	btscanner 0	7.31
46	3	b3b0628	bum_sched	41.26

Figure 20-110. **onstat -g sle** command output

onstat -g smb command: Print sbspaces information

Use the **onstat -g smb** command to display detailed information about sbspaces.

Syntax:

```
➤ onstat -g smb ➤
```

c
e
fdd
lod
s
h
cad
fdd
lod

Command	Explanation
onstat -g smb c	Lists all the chunks in the sbspace.
onstat -g smb e	Lists the entries in the large object header table.
onstat -g smb fdd	Lists the smart-large-object file descriptors.
onstat -g smb lod	Lists the header and entries in the smart-large-object header table.
onstat -g smb s	Lists the sbspace attributes (owner, name, page size, -Df flag settings). Fields with a value of 0 or -1 were not initialized during sbspace creation.
onstat -g smb h	Lists the headers of all smart-large-object table types.
onstat -g smb h cad	Lists the header for the smart-large-object chunk adjunct table.
onstat -g smb h fdd	Lists the header for the smart-large-object file descriptor table.
onstat -g smb h lod	Lists the header for the smart-large-object header table.

Example output for the onstat -g smb c command

Use the **onstat -g smb c** command to monitor the amount of free space in each sbspace chunk, and the size in pages of the user data, metadata, and reserved areas. The **onstat -g smb c** command displays the following information for each sbspace chunk:

- Chunk number and sbspace name
- Chunk size and pathname
- Total user data pages and free user data pages
- Location and number of pages in each user-data and metadata areas

In the following example, chunk 2 of sbspace1 has 2253 used pages (usr pgs) and 2245 free pages (free pg). For the first user-data area Ud1, the starting page offset is 53 and the number of pages is 1126. For the metadata area Md, the starting page offset is 1179 and the number of pages is 194. For the reserved data Ud2, the starting page offset is 1373 and the number of pages is 1127.

Chunk Summary:

```

sbnm 2  chunk 2
chunk:  address  flags   offset   size   orig fr   usr pgs   free pg
        303cf2a8  F-----  0        2500   2253     2253     2245
        path: /usr11/myname/sbspace1

        start pg   npages
Ud1   :    53      1126
Md    :   1179     194
Ud2   :   1373    1127

```

Output for the onstat -g smb s command

The **onstat -g smb s** command displays the storage attributes for all sbspaces in the system:

- sbspace name, flags, owner
- logging status
- average smart-large-object size
- first extent size, next extent size, and minimum extent size

- maximum I/O access time
- lock mode

For more information on the **onstat -g smb** command, see the *IBM Informix Performance Guide*.

onstat -g smx command: Print multiplexer group information

Use the **onstat -g smx** command to display information about the server multiplexer group for servers using SMX.

Syntax:

```

▶▶ onstat -g smx [ses] ▶▶

```

Command	Explanation
onstat -g smx	Displays SMX connection statistics
onstat -g smx ses	Displays SMX session statistics

Example output

```

SMX connection statistics:
SMX control block: 0x47d5e028

  Peer server name: lx1
  SMX connection address: 0x47d60d10
  Encryption status: Disabled
  Total bytes sent: 27055
  Total bytes received: 2006989
  Total buffers sent: 782
  Total buffers received: 7090
  Total write calls: 782
  Total read calls: 7090
  Total retries for write call: 0
  Data compression level: 1
  Data sent: compressed 40760 bytes by 33%
  Data received: compressed 12579324 bytes by 84%

```

Figure 20-111. **onstat -g smx** command output

Output description

SMX control block

SMX control block

Peer server name

Displays the name of the peer server

SMX connection address

Displays the address of the SMX connection

Encryption status

Displays whether encryption is enabled or disabled

Total bytes sent

Displays the total number of bytes sent

Total bytes received
Displays the total number of bytes received

Total buffers sent
Displays the total number of buffers sent

Total buffers received
Displays the total number of buffers received

Total write calls
Displays the total number of write calls

Total read calls
Displays the total number of read calls

Total retries for write call
Displays the total number of retries for write call

Data compression level
Displays the SMX compression level as set by the SMX_COMPRESS configuration parameter

Data sent: compressed x bytes by y%
Displays the uncompressed number of bytes and the compression ratio of the data sent

Data received: compressed x bytes by y%
Displays the uncompressed number of bytes and the compression ratio of the data received

Example Output

```
SMX session statistics:
SMX control block: 0x17c69028
```

Peer name	SMX session address	client type	reads	writes
delhi_sec	19022050	smx Clone Send	6	183

Figure 20-112. `onstat -g smx ses` Output

Output Description

SMX control block
SMX control block

Peer name
Displays the name of the peer server

SMX session address
SMX session address

Client type
Displays type of secondary server

reads Displays the total number of session reads

writes Displays the total number of session writes

onstat -g spi command: Print spin locks with long spins

Use the `onstat -g spi` command to display information about spin locks with long spins.

Syntax:

►—onstat— -g—spi—◄◄

Many resources in the server are accessed by two or more threads. In some of these accesses (such as updating a shared value), the server must guarantee that only one thread is accessing the resource at a time. A *spin lock* is the mechanism used to provide this mutually exclusive access for some resources. With this type of lock, a thread that did not succeed in acquiring the lock on the first try (because another thread was holding it) repeatedly attempts to acquire the lock until it succeeds.

The overhead cost of a spin lock is small, and spin locks are normally used for resources that require mutual exclusion for short periods of time. However, if a spin lock becomes highly contended, the loop-and-retry mechanism can become expensive.

The **onstat -g spi** command is helpful for identifying performance bottlenecks that are caused by highly contended spin locks. This option lists spin locks with waits, those spin locks for which a thread was not successful in acquiring the lock on its first attempt and thus had to loop and re-attempt.

Example output

Spin locks with waits:

Num Waits	Num Loops	Avg Loop/Wait	Name
114	117675	1032.24	lockfr3
87	256461	2947.83	fast mutex, lockhash[832]
1	11	11.00	fast mutex, 1:bhash[16668]
4	51831	12957.75	fast mutex, 1:lru-4
1	490	490.00	fast mutex, 1:bf[994850] 0xe00002 0x14eb32000

Figure 20-113. **onstat -g spi** command output

Output description

Num Waits (decimal)

Total number of times a thread waited for this spin lock.

Num Loops (decimal)

Total number of attempts before a thread successfully acquired the spin lock.

Avg Loop/Wait (floating point)

Average number of attempts needed to acquire the spin lock. Computed as Num Loops / Num Waits.

Name (string)

Uses the following codes to name the spin lock

lockfr The lock free list. The number after **lockfr** is the index into the lock free list array.

lockhash[]

The lock hash bucket. The field inside the brackets is the index into the lock hash bucket array.

:bhash []

The buffer hash bucket. The field before the colon is the buffer pool index; the field inside the brackets after **bhash** is the index into the buffer hash bucket array.

:lru-

The LRU latch. The field before the colon is the buffer pool index; the field after **lru-** identifies the buffer chain pairs that are being used.

:bfl[]

The buffer latch. The field before the colon is the buffer pool index; the field inside the brackets after **bf** is the position of buffer in the buffer array. The next two fields are the partition number and the page header address in memory for the buffer in hex form.

onstat -g sql command: Print SQL-related session information

Use the **onstat -g sql** command to display SQL-related information about a session.

By default, only the DBSA can view **onstat -g sql** syssqltrace information. However, when the UNSECURE_ONSTAT configuration parameter is set to 1, all users can view this information.

Syntax:

►►—onstat— -g—sql—*sessionid*—◄◄

You can specify one of the following invocations.

Invocation

Explanation

onstat -g sql

Displays a one line summary for each session

onstat -g sqlsessionid

Displays SQL information for a specific session

Note: Encrypted passwords and password hint parameters in encryption functions are not shown. The following figure displays an encrypted password in the Last parsed SQL statement field.

```
onstat -g sql 22
```

Sess	SQL	Current	Iso Lock	SQL	ISAM	F.E.	Current
Id	Stmt type	Database	Lvl Mode	ERR	ERR	Vers Explain	Role
22	-	test	CR Not Wait	0	0	9.03 Off	hr

Last parsed SQL statement :

```
select id, name, decrypt_char(ssn, 'XXXXXXXXXX') from emp
```

Figure 20-114. **onstat -g sql** command output

Output description

Sess id The session identifier

SQL Stmt type

The type of SQL statement

Current Database

Name of the current database of the session

ISO Lvl

Isolation level

DR Dirty Read

CR Committed Read

CS Cursor Stability

DRU Dirty Read, Retain Update Locks

CRU Committed Read, Retain Update Locks

CSU Cursor Stability, Retain Update Locks

LC Committed Read, Last Committed

LCU Committed Read Last Committed with Retain Update Locks

RR Repeatable Read

NL Database Without Transactions

Lock mode

Lock mode of the current session

SQL Error

SQL error number encountered by the current statement

ISAM Error

ISAM error number encountered by the current statement

F.E. Version

The version of the SQLI protocol used by the client program

Explain

SET EXPLAIN setting

Current Role

Role of the current user

onstat -g src command: Patterns in shared memory

Use the **onstat -g src** command to search for patterns in shared memory.

Syntax:

►►—onstat— -g—src—*pattern*—*mask*—◄◄

Example output

The following example shows output for the **onstat -g src***pattern mask* command where *pattern* = 0x123 and *mask* = 0xffff.

```
Search Summary:
addr           contents
00000000ad17a50: 01090000 00000000 00000000 00000123 .....#
00000000ad7dec0: 00000001 014e3a0c 00000000 0ade0123 .....N:.#
```

Figure 20-115. **onstat -g src** command output

Output description

addr (hexadecimal)

Address in shared memory where search pattern is found

contents (hexadecimal)

Contents of memory at given address

onstat -g ssc command: Print SQL statement occurrences

Use the **onstat -g ssc** command to monitor the number of times that the database server reads the SQL statement in the cache.

By default, only the DBSA can view **onstat -g ssc** syssqltrace information. However, when the UNSECURE_ONSTAT configuration parameter is set to 1, all users can view this information.

Syntax:



The **all** option reports the *key-only* cache entries as well as the fully cached statements. If the value in the **hits** column is less than the STMT_CACHE_HITS value, that entry is a *key-only* cache entry. For more information, see memory utilization in the *IBM Informix Performance Guide*.

The **pool** option reports usage of all memory pools for the SQL statement cache. The output displays information on the name, class, address, and total size of the memory pools. For more information, see improving query performance in the *IBM Informix Performance Guide*.

Example output

```

Statement Cache Summary:
#lrus   currsz  maxsz  Poolsize #hits  nolimit
4       117640 524288 139264   0       1
Statement Cache Entries:
lru hash ref_cnt hits flag heap_ptr      database      user
-----
0 262      0   7  -F aad8038      sscsi007      admin
  INSERT INTO ssc1 ( t1_char , t1_short , t1_key , t1_float , t1_smallfloat
    , t1_decimal , t1_serial ) VALUES ( ? , ? , ? , ? , ? , ? , ? )
0 127      0   9  -F b321438      sscsi007      admin
  INSERT INTO ssc2 ( t2_char , t2_key , t2_short ) VALUES ( ? , ? , ? )
1 134      0  15  -F aae0c38      sscsi007      admin
  SELECT t1_char , t1_short , t1_key , t1_float , t1_smallfloat ,
    t1_decimal , t1_serial FROM ssc1 WHERE t1_key = ?
1 143      0   3  -F b322c38      sscsi007      admin
  INSERT INTO ssc1 ( t1_char , t1_key , t1_short ) SELECT t2_char , t2_key
    + ? , t2_short FROM ssc2
2  93      0   7  -F aae9838      sscsi007      admin
  DELETE FROM ssc1 WHERE t1_key = ?
2 276      0   7  -F aaefc38      sscsi007      admin
  SELECT count ( * ) FROM ssc1
2 240      1   7  -F b332838      sscsi007      admin
  SELECT COUNT ( * ) FROM ssc1 WHERE t1_char = ? AND t1_key = ? AND
    t1_short = ?
3  31      0   7  -F aaec038      sscsi007      admin
  SELECT count ( * ) FROM ssc1 WHERE t1_key = ?
3  45      0   1  -F b31e438      sscsi007      admin
  DELETE FROM ssc1
3 116      0   0  -F b362038      sscsi007      admin
  SELECT COUNT ( * ) FROM ssc1
Total number of entries: 10.

```

Figure 20-116. **onstat -g ssc** command output

Output description - Statement Cache Summary section

#lrus Number of least recently used queues (LRUS)

currsz

Current cache size

maxsz

Limit on total cache memory

Poolsize

Total pool size

#hits

The number of hits before insertion. This number equals the value of the STMT_CACHE_HITS configuration parameter

nolimit

The value of the STMT_CACHE_NOLIMIT configuration parameter

Output description - Statement Cache Entries section

The Statement Cache Entries section shows the entries that are fully inserted into the cache.

lru

The index of lru queue to which the cache entry belongs

hash

Hash values of cached entry

ref_count

Number of threads referencing the statement

hits Number of times a statement matches a statement in the cache. The match can be for a key-only or fully cached entry.

flag Cache entry flag -F indicates the statement is fully cached -D indicates the statement is dropped

heap_ptr Address of memory heap for cache entry

onstat -g stk command: Print thread stack

Use the **onstat -g stk *tid*** command to display the stack of the thread specified by thread ID.

Syntax:

►► onstat — -g — stk — *tid* ————— ◄◄

Example output

```
Stack for thread: 2 adminthd
base: 0x000000010aad5028
len: 33280
pc: 0x00000001002821e8
tos: 0x000000010aad621
state: running
vp: 2

0x1002821e8 oninit :: yield_processor + 0x260 sp=0x10aadce20(0x10ac834d0, 0x0, 0x1,
0x100000000, 0xc8a000, 0x100c8a000)
0x100274e38 oninit :: wake_periodic + 0xdc sp=0x10aadced0 delta_sp=176(0x41b0, 0xc7a024bc,
0x0, 0x41c4, 0x10aacf598, 0x90)
0x100274fcc oninit :: admin_thread + 0x108 sp=0x10aadcf80 delta_sp=176(0x0, 0x2328,
0xd26c00, 0x5, 0xc8a000, 0x156c)
0x1002484ec oninit :: startup + 0xd8 sp=0x10aadd050 delta_sp=208(0xa, 0x10aad47d0,
0x10aad47d0, 0x100db1988, 0xd1dc00, 0x1)
```

Figure 20-117. **onstat -g stk *tid*** command output

onstat -g stm command: Print SQL statement memory usage

Use the **onstat -g stm** command to display the memory that each prepared SQL statement uses.

By default, only the DBSA can view **onstat -g stm** syssqltrace information. However, when the UNSECURE_ONSTAT configuration parameter is set to 1, all users can view this information.

Syntax:

►► onstat — -g — stm ————— ◄◄

To display the memory for only one session, specify the session ID in the **onstat -g stm** command.

Example output

```
session 65 -----
sdblock heap sz statement ('*' = Open cursor)
aad8028 16544 SELECT COUNT ( * ) FROM ssc1 WHERE t1_char = ?
AND t1_key = ? AND t1_short = ?
```

Figure 20-118. **onstat -g stm** command output

Output description

sdblock Address of the statement descriptor block

heap sz Size of the statement memory heap

statement

Query text

onstat -g stq command: Print queue information

Use the **onstat -g stq** command to display information about the queue.

Syntax:

►► onstat — -g — stq — session ————— ►►

To view queue information for a particular session specify the *session* option. Omit the *session* option to view queue information for all sessions.

Example output

```
Stream Queue: (session 25 cnt 4) 0:db12400 1:db18400 2:dcf0400 3:dcf6400
Full Queue: (cnt 2 waiters 0) 0:0 1:db12400
Empty Queue: (cnt 0 waiters 0)
```

Figure 20-119. **onstat -g stq** command output

Output description

session Session id

cnt Number of stream queue buffers

waiters Number of threads waiting for the stream queue buffer

onstat -g sts command: Print stack usage for each thread

Use the **onstat -g sts** command to display information about the maximum and current stack use for each thread.

Syntax:

►► onstat — -g — sts ————— ►►

Example output

Stack usage:						
TID	Total	Max bytes	%	Current bytes	%	Thread Name
2	32768	3124	9	3079	9	adminthd
3	32768	2870	8	2871	8	childthd
5	32768	14871	45	2871	8	Cosvr Avail Mgr
6	32768	2870	8	2871	8	dfm_svc
7	131072	3190	2	3191	2	xmf_svc
9	32768	3126	9	3127	9	xtm_svcc
10	32768	3580	10	3335	10	xtm_svc
11	32768	3238	9	3239	9	cfgmgr_svc
12	32768	6484	19	2871	8	lio vp 0
14	32768	6484	19	2871	8	pio vp 0
16	32768	6484	19	2871	8	aio vp 0
18	131072	10391	7	2871	2	msc vp 0
20	32768	4964	15	2871	8	fifo vp 0
22	32768	4964	15	2871	8	fifo vp 1
24	32768	6028	18	2871	8	aio vp 1
26	32768	5444	16	2951	9	dfmxpl_svc
27	32768	2886	8	2887	8	sch_svc
28	32768	7812	23	5015	15	rqm_svc
29	32768	7140	21	3079	9	sm_poll
30	32768	11828	36	6439	19	sm_listen
31	32768	2870	8	2871	8	sm_discon
32	32768	14487	44	4055	12	main_loop()
33	32768	4272	13	2903	8	flush_sub(0)
34	32768	2902	8	2903	8	flush_sub(1)
35	32768	2870	8	2871	8	btscanner 0
36	32768	3238	9	3239	9	aslogflush
37	32768	3055	9	2887	8	bum_local
38	32768	3238	9	3239	9	bum_rcv
39	32768	4902	14	4903	14	onmode_mon
42	32768	4964	15	2871	8	lio vp 1
44	32768	5136	15	2871	8	pio vp 1

Figure 20-120. **onstat -g sts** command output

onstat -g sym command: Print symbol table information for the oninit utility

Use the **onstat -g sym** command to display symbol table information for the **oninit** utility.

Syntax:

►► onstat — -g — sym ————— ◀◀

Example output

The following example shows the first few lines from the output:

```
Table for oninit has 23378 entries
Initial value for -base-: 0x0
0x3451e0 _start
0x345300 .ld_int
0x345348 .ld_lli
0x3453dc .ld_float
0x345428 .ld_double
0x3454c4 .st_int
0x3454fc .st_lli
0x34556c .st_float
0x3455c0 .st_double
0x34565c .st_float_foreff
0x345694 .st_double_foreff
0x345718 main
0x34c2ac get_cfgfile
0x34c2fc is_server_alias
```

Figure 20-121. **onstat -g sym** command output

Output description

The **onstat -g sym** command displays the relative in-memory address and name of symbols (functions and variables) in the **oninit** utility.

onstat -g sync command: Print ER synchronization status

Use the **onstat -g sync** command to display the synchronization status when Enterprise Replication is used.

The **onstat -g sync** command is used primarily as a debugging tool and most often by Technical Support.

Syntax:

►►—onstat— -g—sync—————◄◄

Example output

Prim Repl	Sync Source	St.	Shadow Repl	Flag	Stat	Block Num	EndBlk Num
655361	20	0	1310729	2	0	592	600

Figure 20-122. **onstat -g sync** command output

Output description

Prim Repl

Replicate number of the replicate being synchronized

Sync Source

Source server of the sync

St

Sync replicate state

Shadow Repl

The shadow replicate used to perform the sync

Flag Internal flags:

- 0x02 = external sync
- 0x04 = shutdown request has been issued
- 0x08 = abort has occurred
- 0x010 = a replicate stop has been requested
- 0x020 = shadow or primary replicate has been deleted

Stat Resync job state

Block num
Last block applied on targets (on source always 0)

EndBlock Num
Last block in resync process. Marks the end of the sync scan on the target.
A value of -2 indicates that the scan is still in progress, and the highest block number is not yet known.

Additional fields for forwarded rows:

ServID Server where forwarded row originated

fwdLog ID
Originator's log ID of the forwarded row

fwdLog POS
Originator's log position of the forwarded row

endLog ID
Operation switches back to normal at this point

endLog POS
Operation switches back to normal at this log position

complete flag
Set to 1 after normal processing resumes for the originating source.

onstat -g tpf command: Print thread profiles

Use the **onstat -g tpf** command to display thread profiles.

Syntax:

►► onstat — -g tpf — *tid* ————— ►►

Specify the *tid* thread ID to print the profile for a specific thread. Set *tid* to 0 to display the profiles for all of the threads.

Example output

```
onstat -g tpf 945

Thread profiles
tid lkreqs lkw dl to lgrs isrd iswr isrw isdl isct isrb lx bfr bfw lsus lsmx seq
945 1969 0 0 0 6181 1782 2069 13 0 0 0 0 16183 7348 743580 0 6
```

Figure 20-123. **onstat -g tpf** command output

Output description

<i>tid</i>	Thread ID
<i>lkreqs</i>	Lock requests
<i>lkw</i>	Lock waits
<i>dl</i>	Deadlocks
<i>to</i>	Remote deadlock timeout
<i>lgrs</i>	Log records
<i>isrd</i>	Number of reads
<i>iswr</i>	Number of writes
<i>isrw</i>	Number of rewrites
<i>isdl</i>	Number of deletes
<i>isct</i>	Number of commits
<i>isrb</i>	Number of rollbacks
<i>lx</i>	Long transactions
<i>bfr</i>	Buffer reads
<i>bfw</i>	Buffer writes
<i>lsus</i>	Log space currently used
<i>lsmx</i>	Max log space used
<i>seq</i>	Sequence scans

onstat -g ufr command: Print memory pool fragments

Use the **onstat -g ufr** command to display a list of the fragments that are currently in use in the specified memory pool.

This command requires an additional argument to specify either a pool name or session ID whose memory pool information is to be displayed. Each session is allocated a memory pool with the same name as the session ID. Use the **onstat -g mem** command to identify the pool name and the **onstat -g ses** command to identify the session ID.

Syntax:

```
►► onstat -g ufr pool name  
sessionid ►►
```

Memory pools are broken into fragments for various uses. With the **onstat -g ufr** command it is possible to see a list of these fragments showing their respective sizes in bytes and the type of information they contain. The information provided is generally used by Technical Support to assist in the analysis of a reported problem.

Example output for a specified pool name

```
Memory usage for pool name global:
size      memid
1736      overhead
23544     mcbmsg
72        messages
33112     osend
25432     rsam
88        shmbklist
5170664   net
```

Figure 20-124. **onstat -g ufr global** command output for a specified pool name

Example output for a specified session ID

The following example shows the output for session ID 6.

```
Memory usage for pool name 6:
size      memid
3256      overhead
144       scb
2968      ostcb
18896     sqscb
3312      opentable
72        sql
808       filetable
352       fragman
552       hashfiletab
1584      gentcb
12096     log
2960      sqtcb
2928      osend
720       keys
224       rdahead
16248     temprec
```

Figure 20-125. **onstat -g ufr** command output for a specified session ID

Output description

size (decimal)

Size, in bytes, of the pool fragment.

memid (string)

Name of the pool fragment.

onstat -g vpcache command: Print CPU VP memory block cache statistics

Use the **onstat -g vpcache** command to display information about CPU VP memory block cache statistics.

Syntax:

```
▶▶ onstat — -g vpcache —▶▶
```

Example output

CPU VP memory block cache statistics - 4096 byte blocks						
Number of 4096 byte memory blocks requested for each CPU VP:250						
vpid	pid	Blocks held	Hit percentage	Free cache		
1	7889	193	77.4 %	21.9 %		
Current VP total allocations from cache:				0		
size	cur blks	alloc	miss	free	drain	
1	30	13	4	43	0	
2	12	3	0	9	0	
3	42	7	0	21	0	
4	4	0	0	1	0	
5	0	0	0	0	0	
6	0	0	0	0	0	
7	0	0	0	0	0	
8	8	0	0	1	0	
9	63	0	0	7	0	
10	0	0	0	0	0	
11	0	0	0	0	0	
12	0	0	0	0	0	
13	0	0	0	0	0	
14	0	0	0	0	0	
15	0	0	0	0	0	
16	0	0	0	0	0	
17	34	1	3	3	0	
18	0	0	0	0	0	
19	0	0	0	0	0	
20	0	0	0	0	0	
21	0	0	0	0	0	
22	0	0	0	0	0	
23	0	0	0	0	0	
24	0	0	0	0	0	
25	0	0	0	0	0	
26	0	0	0	0	0	
27	0	0	0	0	0	
28	0	0	0	0	0	
29	0	0	0	0	0	
30	0	0	0	0	0	
31	0	0	0	0	0	
32	0	0	0	0	0	

Figure 20-126. **onstat -g vpcache** command output

Output description

size Is the size of the memory blocks in 4096 byte blocks

cur blks

Is the current number of 4096 blocks, a multiple of *size*

alloc Is the number of times a requestor received a block of this size

miss Is the number of times a block was requested but none were available

free Is the number of times a memory block was placed into the cache

drain Is the number of times an aged block was forced out to make room for another block

onstat -g wai command: Print wait queue thread list

Use the **onstat -g wai** command to display a list of the threads in the system that are currently in the wait queue and not currently executing. The output is sorted by thread ID.

Syntax:

►► onstat — -g—wai —————►►

Example output

```
Waiting threads:
tid      tcb          rstcb          prty status          vp-      name
2        46b1ea40          0              1    IO Idle          5lio     lio vp 0
3        46b3dc58          0              1    IO Idle          6pio     pio vp 0
4        46b5dc58          0              1    IO Idle          7aio     aio vp 0
5        46b7cc58          0              1    IO Idle          8msc     msc vp 0
6        46b1ed10      460f5028        1    sleeping secs: 1   3cpu     main_loop()
9        46d0d6e0          0              1    sleeping forever  1cpu     soctcplst
10       46d70b48          0              1    sleeping forever  3cpu     sm_listen
11       46e5d9a0          0              1    sleeping secs: 1   3cpu     sm_discon
12       46e5dc70      460f5820        1    sleeping secs: 1   3cpu     flush_sub(0)
13       46e8a5a8      460f6018        1    sleeping secs: 1   3cpu     aslogflush
14       46fe8148      460f6810        1    sleeping secs: 41  3cpu     btscanner_0
15       46fe84a8          0              1    IO Idle          10aio    aio vp 1
16       46fe8778      460f7008        1    sleeping secs: 1   1cpu     onmode_mon
36       47531960      460f7ff8        1    sleeping secs: 253  3cpu     dbScheduler
37       47531c30      460f87f0        1    sleeping forever   4cpu     dbWorker1
38       47491028      460f7800        1    sleeping forever   4cpu     dbWorker2
```

Figure 20-127. **onstat -g wai** command output

Output description

tid (decimal)

Thread ID

tcb (hex)

In-memory address of the thread control block

rstcb (hex)

In-memory address of the RSAM thread control block

prty (decimal)

Thread priority. Higher numbers represent higher priorities

status (string)

Current status of the thread

vp- (decimal and string)

Virtual processor integer ID of the VP on which the thread last ran, concatenated with the name of the VP upon which the thread runs

name (string)

Name of the thread

onstat -g wmx command: Print all mutexes with waiters

Use the **onstat -g wmx** command to display all of the mutexes with waiters.

Syntax:

►► onstat — -g — wmx ————— ◄◄

Example output

```
Mutexes with waiters:
mid      addr      name      holder  lkcnt  waiter  waittime
```

Figure 20-128. **onstat -g wmx** command output

onstat -g wst command: Print wait statistics for threads

Use the **onstat -g wst** command to display the wait statistics for the threads within the system.

The WSTATS configuration parameter must be set to 1 to enable wait statistics collection. For more information, see “WSTATS configuration parameter” on page 1-163.

Syntax:

►► onstat — -g — wst ————— ◄◄

Example output

Figure 20-129. **onstat -g wst** command output

```
Version 11.70.F -- On-Line -- Up 18:52:59 -- 78856 Kbytes
name  tid  state      n      avg(us)  max(us)
msc vp 0 5  ready      6         9        17
msc vp 0 5  run        6       1107     2215
msc vp 0 5  IO Idle    5     2985.9s  1496.1s

main_loo 7  IO Wait    55       6496    16725
main_loo 7  yield time 44929     1.2s    343.1s
main_loo 7  ready     44998    206085    343.1s
main_loo 7  run       44985         5        436

...

sqlxec 63  IO Wait     2       1118     2165
sqlxec 63  other cond   6     34237   204142
sqlxec 63  ready       9         7         16
sqlxec 63  run        7       1.1s     7.7s
```

Output description

name (string)

Thread name

tid (decimal)

Thread ID

state (string)

State the thread waited in for this line of output. A single thread can have multiple lines of output if it has waited in more than one different state. Values that can appear in the state include:

chkpt cond: The thread waited for a checkpoint condition.
cp mutex: The thread waited for checkpoint mutex to become available.
deadlock mutex: The thread waited for a deadlock mutex to become available.
empty Q: The thread waited for an empty buffer on a queue.
fork: The thread waited for a child thread to run.
full Q: The thread waited for a full buffer on a queue.
IO Idle: The I/O thread was idle.
IO Wait: The thread yielded while waiting for I/O completion.
join wait: The thread waited for another thread to exit.
lock mutex: The thread waited for lock mutex to become available.
lockfree mutex: The thread waited for a lock-free mutex to become available.
logflush: Logical log flushing occurred.
log mutex: The thread waited for logical log mutex to become available.
logcopy cond: The thread waited for logical log copy condition.
logio cond: The thread waited for a logical log condition.
lrus mutex: The thread waited for a buffer LRU mutex to become available.
misc: The thread waited for a miscellaneous reason.
other cond: The thread waited for an internal condition.
other mutex: The thread waited for an internal system mutex to become available.
other yield: The thread yielded for an internal reason.
OS read: The thread waited for an operating system read call to complete.
OS write: The thread waited for an operating system write call to complete.
ready: The thread was ready to run.
run: The thread ran.
sort io: The thread waited for sort I/O completion.
vp mem sync: The thread waited for synchronization of virtual processor memory.
yield bufwait: The thread yielded while waiting for a buffer to become available.
yield 0: The thread yielded with an immediate timeout.
yield time: The thread yielded with a timeout.
yield forever: The thread yielded and will stay that way until it wakes up.

n (decimal)

Number of times the thread waited in this state

avg(us) (floating point)

Average user time the thread spent waiting in this state per wait occurrence. Time is in microseconds; an *s* after the value indicates user time in seconds.

max(us) (floating point)

Maximum user time the thread spent waiting in this state for a single wait occurrence. Time is in microseconds; an *s* after the value indicates user time in seconds.

Related reference:

“**onstat -g ath** command: Print information about all threads” on page 20-54

onstat -G command: Print TP/XA transaction information

Use the **onstat -G** command to display information about global transactions generated through the TP/XA library.

Syntax:

►► **onstat** — **-G** —————►►

Example output

```
Global Transaction Identifiers
address  flags  isol  timeout  fID      gtl  bql  data
ae35e34  -LR-G  COMMIT  0        4478019  16  48  438709F23076254C80F33A62B
AF4CF763C1BCFFAD7AE0243AA5CE243FA5381C903AA9F52A1546044992C5A7BC03582E77999EFBA7
25D3D40BDAF37404D9DAFF1
ae3623c  AL--G  COMMIT  0        4478019  16  48  438709F23076254C80F33A62B
AF4CF763C1BCFFAD7AE0243AA5CE243FA5381C903AA9F52A1546044992C5A7BC03582E77999EFBA7
25D3D40BDAF37404D9DA000
2 active, 128 total
```

Figure 20-130. **onstat -G** command output

For a tightly coupled transaction, all branches will share the same transaction address shown in the address column.

Output description**address**

Transaction address

flags

The flag codes for position 1 (current transaction state):

- A** User thread attached to the transaction
- S** TP/XA suspended transaction
- C** TP/XA waiting for rollback

The flag codes for position 2 (transaction mode):

- T** Tightly-coupled mode (MTS)
- L** Loosely-coupled mode (default mode)

The flag codes for position 3 (transaction stage):

- B** Begin work
- P** Distributed query prepared for commit
- X** TP/XA prepared for commit
- C** Committing or committed
- R** Rolling back or rolled back
- H** Heuristically rolling back or rolled back

The flag code for position 4:

- X** XA DataSource global transaction

The flag codes for position 5 (type of transaction):

- G** Global transaction
- C** Distributed query coordinator
- S** Distributed query subordinate
- B** Both distributed query coordinator and subordinate
- M** Redirected global transaction

isol Transaction isolation level

timeout
Transaction lock timeout

fID Format ID

gtl Global transaction ID length

bql Branch qualifier length

data Transaction-specific data

onstat -h command: Print buffer header hash chain information

Use the **onstat -h** command to display information about the buffer header hash chains (sometimes called "hash buckets") that are used to access pages in each buffer pool.

Syntax:

►► onstat — -h ————— ◀◀

Example output

The output is displayed in the form of a numeric histogram of chain lengths, with summary information for each buffer pool. All numeric values in the output are decimal. Shorter hash chains enable requested buffers to be located more quickly by the server, because on average it will need to check fewer buffer headers on a target chain to find the target buffer.

The page size of the buffer pool in bytes is shown as a header to the output for each buffer pool. The histogram and summary information are then presented for that buffer pool.

```
Buffer pool page size: 2048

buffer hash chain length histogram
# of chains      of len
      3423             0
      4546             1
       223             2
      8192  total chains
      4992  hashed buffs
      5000  total buffs

Buffer pool page size: 4096

buffer hash chain length histogram
# of chains      of len
      707             0
      315             1
       2              2
     1024  total chains
      319  hashed buffs
     1000  total buffs
```

Figure 20-131. **onstat -h** command output

Output description

Histogram Information on Hash Chains

The histogram information has a row for each buffer hash chain length that presently exists in the system. Each row has two columns:

of chains

Number of hash chains of the given length

of len

Length of these chains

Summary Information Per Buffer Pool

total chains

Number of hash chains that exist for this buffer pool

hashed buffs

Number of buffer headers currently hashed into the hash chains for this buffer pool

total buffs

Total number of buffers in this buffer pool

onstat -i command: Initiate interactive mode

Use the **onstat -i** command to put the **onstat** utility in the interactive mode.

Syntax:

```

▶▶ onstat -i [r seconds] [rz seconds]

```

In interactive mode, you can enter multiple **onstat** options per session, but only one at a time. An **onstat** prompt appears and allows you to enter an option.

Important: In interactive mode, do not precede the option with a dash.

Additional options

Two additional options, **onstat r seconds** and **onstat rz seconds**, are available in interactive mode. The **onstat r seconds** option is similar to the current **onstat -r seconds** option, which repeatedly generates a display. If an administrator executes **onstat r seconds** at the interactive-mode prompt, the prompt changes to reflect the specified interval in seconds and reappears, waiting for the next command. In the following example, the display generated by the next command repeats every three seconds:

```
onstat> r 3
onstat[3]>
```

The **onstat rz seconds** option enables you to repeat the next command as specified and set all profile counters to 0 between each execution.

Terminating interactive mode or repeating sequence

To terminate the interactive mode, press CTRL-d.

To terminate a repeating sequence, press CTRL-c.

onstat -j command: Provide onpload status information

Use the **onstat -j** command to provide information about the status of an **onpload** job.

The **onstat -j** command provides an interactive mode that is analogous to the **onstat -i** command.

Syntax:

►►—onstat— -j—►►

When the **onpload** starts, it writes a series of messages to **stdout** or to a log file. The following lines show a typical **onpload** log file:

```
Mon Jul 23 16:11:30 2010
```

```
SHMBASE      0x4400000
CLIENTNUM    0x49010000
Session ID 1
```

```
Load Database -> cnv001
Load Table    -> cnv001a
Load File     -> testrec.dat
Record Mapping -> cnv001a
```

```
Database Load Completed -- Processed 50 Records
Records Inserted-> 50
Detected Errors--> 0
Engine Rejected--> 0
```

```
Mon Jul 23 16:11:37 2010
```

Output description

The two lines that start with SHMBASE and CLIENTNUM provide the information that you need to locate shared memory for an instance of **onpload**. The **oninit** process has similar values stored in the **\$ONCONFIG** file. When you use the

onstat utility to gather information about the **oninit** process, the **onstat** utility uses information from **\$INFORMIXDIR/etc/\$ONCONFIG** file to locate shared memory. When you use **onstat** to gather information about **onpload**, you must give the **onstat** utility the name of a file that contains SHMBASE and CLIENTNUM information.

Typically the file that contains the SHMBASE and CLIENTNUM information is the log file. For example, if the **onpload** log file is **/tmp/cnv001a.log**, you can enter the following command:

```
onstat -j /tmp/cnv001a.log
```

The previous command causes the **onstat** utility to attach to **onpload** shared memory and to enter interactive mode. You can then enter ? or any other pseudo request to see a usage message displayed. An example follows:

```
onstat> ?
Interactive Mode: One command per line, and - are optional.
    -rz      repeat option every n seconds (default: 5) and
              zero profile counts
MT COMMANDS:
    all      Print all MT information
    ath      Print all threads
    wai      Print waiting threads
    act      Print active threads
    rea      Print ready threads
    sle      Print all sleeping threads
    spi      print spin locks with long spins
    sch      print VP scheduler statistics
    lmx      Print all locked mutexes
    wmx      Print all mutexes with waiters
    con      Print conditions with waiters
    stk <tid> Dump the stack of a specified thread
    glo      Print MT global information
    mem <pool name|session id> print pool statistics.
    seg      Print memory segment statistics.
    rbm      print block map for resident segment
    nbm      print block map for non-resident segments
    afr <pool name|session id> Print allocated pool fragments.
    ffr <pool name|session id> Print free pool fragments.
    ufr <pool name|session id> Print pool usage breakdown
    iov      Print disk IO statistics by vp
    iof      Print disk IO statistics by chunk/file
    ioq      Print disk IO statistics by queue
    iog      Print AIO global information
    iob      Print big buffer usage by IO VP
    sts      Print max and current stack sizes
    qst      print queue statistics
    wst      print thread wait statistics
    jal      Print all Pload information
    jct      Print Pload control table
    jpa      Print Pload program arguments
    jta      Print Pload thread array
    jmq      Print Pload message queues, jms for summary only
onstat>
```

Most of the options are the same as those that you use to gather information about Informix, with the following exceptions:

```
jal      Print all Pload information
jct      Print Pload control table
jpa      Print Pload program arguments
jta      Print Pload thread array
jmq      Print Pload message queues, jms for summary only
```

These options apply only to **onpload**. You can use the **onstat -j** command to check the status of a thread, locate the VP and its PID, and then attach a debugger to a particular thread. The options for the **onstat** utility that do not apply to **onpload** are not available (for example, **onstat -g ses**).

onstat -k command: Print active lock information

Use the **onstat -k** command to print information about active locks, including the address of the lock in the lock table.

Syntax:

➤ onstat — -k ————— ➤

Example output

Note: The maximum number of locks available is specified as LOCKS in the ONCONFIG file.

```
Locks
address  wtlist  owner    lklist   type     tblsnum  rowid    key#/bsiz
a095f78  0        a4d9e68  0        HDR+S    100002   203      0
1 active, 2000 total, 2048 hash buckets, 0 lock table overflows
```

Figure 20-132. **onstat -k** command output

Output description

- address* Is the address of the lock in the lock table
- If a user thread is waiting for this lock, the address of the lock appears in the **wait** field of the **onstat -u** (users) output.
- wtlist* Is the first entry in the list of user threads that is waiting for the lock, if there is one
- owner* Is the shared-memory address of the thread that is holding the lock
- This address corresponds to the address in the **address** field of **onstat -u** (users) output. When the *owner* value is displayed in parenthesis, it represents the shared memory address of a transaction structure. This scenario is possible only when a lock is allocated for a global transaction. This address corresponds to the address field of the output for **onstat -G**.
- lklist* Is the next lock in a linked list of locks held by the owner just listed
- type* Uses the following codes to indicate the type of lock:
- | | |
|------------|------------------|
| HDR | Header |
| B | Bytes |
| S | Shared |
| X | Exclusive |
| I | Intent |
| U | Update |
| IX | Intent-exclusive |

IS Intent-shared
SIX Shared, intent-exclusive

tblsnum

Is the tblspace number of the locked resource. If the number is less than 10000, it indicates Enterprise Replication pseudo locks.

rowid

Is the row identification number

The rowid provides the following lock information:

- If the rowid equals zero, the lock is a table lock.
- If the rowid ends in two zeros, the lock is a page lock.
- If the rowid is six digits or fewer and does not end in zero, the lock is probably a row lock.
- If the rowid is more than six digits, the lock is probably an index key-value lock.

key#/bsiz

Is the index key number, or the number of bytes locked for a VARCHAR lock

If this field contains 'K-' followed by a value, it is a key lock. The value identifies which index is being locked. For example, K-1 indicates a lock on the first index defined for the table.

onstat -l command: Print physical and logical log information

Use the **onstat -l** command to display information about the physical logs, logical logs, and temporary logical logs.

Syntax:

►► onstat -l ◀◀

Example Output

```

Physical Logging
Buffer bufused  bufsize  numpages numwrits pages/io
P-1  0          16       716      55      13.02
      phybegin      physize phypos   phyused %used
      1:263         500     270     0       0.00

Logical Logging
Buffer bufused  bufsize  numrecs  numpages numwrits recs/pages pages/io
L-3  0          16      42169   2872    1043    14.7      2.8
      Subsystem    numrecs  Log Space used
      OLDRSAM      42169   4436496

address  number  flags   uniqid  begin          size  used  %used
a517f70  1         U-B----  1       1:763          500   500  100.00
a517fb0  2         U-B----  2       1:1263         500   500  100.00
a40daf0  3         U-B----  3       1:1763         500   500  100.00
a40db30  4         U-B----  4       1:2263         500   500  100.00
a40db70  5         U-B----  5       1:2763         500   500  100.00
a40dbb0  6         U---C-L  6       1:3263         500   372  74.40
a40dbf0  7         A-----  0       1:3763         500    0   0.00
a40dc30  8         A-----  0       1:4263         500    0   0.00
8 active, 8 total

```

Figure 20-133. **onstat -l** command output

Output description for the physical log files

The first section of the display describes the physical-log configuration:

buffer Is the number of the physical-log buffer

bufused

Is the number of pages of the physical-log buffer that are used

bufsize Is the size of each physical-log buffer in pages

numpages

Is the number of pages written to the physical log

numwrits

Is the number of writes to disk

pages/io

Is calculated as $numpages/numwrits$

This value indicates how effectively physical-log writes are being buffered.

phybegin

Is the physical page number of the beginning of the log

physize Is the size of the physical log in pages

phypos Is the current position in the log where the next log-record write is to occur

phyused

Is the number of pages used in the log

%used Is the percent of pages used

The second section of the **onstat -l** command output describes the logical-log configuration:

buffer Is the number of the logical-log buffer

bufused Is the number of pages used in the logical-log buffer

bufsize Is the size of each logical-log buffer in pages

numrecs Is the number of records written

numpages Is the number of pages written

numwrits Is the number of writes to the logical log

recs/pages Is calculated as $\text{numrecs}/\text{numpages}$

You cannot affect this value. Different types of operations generate different types (and sizes) of records.

pages/io is calculated as $\text{numpages}/\text{numwrits}$

You can affect this value by changing the size of the logical-log buffer (specified as LOGBUFF in the ONCONFIG file) or by changing the logging mode of the database (from buffered to unbuffered, or vice versa).

The following fields are repeated for each logical-log file:

address Is the address of the log-file descriptor

number Is logid number for the logical-log file

The logid numbers might be out of sequence because either the database server or administrator can insert a log file in-line.

flags Provides the status of each log as follows:

- A** Newly added (and ready to use)
- B** Backed up
- C** Current logical-log file
- D** Marked for deletion

To drop the log file and free its space for reuse, you must perform a level-0 backup of all storage spaces

- F** Free, available for use
- L** The most recent checkpoint record
- U** Used

uniqid Is the unique ID number of the log

begin Is the beginning page of the log file

size Is the size of the log in pages

used Is the number of pages used

%used Is the percent of pages used

active Is the number of active logical logs

total Is the total number of logical logs

Output description for temporary logical log files

The database server uses *temporary logical logs* during a warm restore because the permanent logs are not available then. The following fields are repeated for each temporary logical-log file:

address Is the address of the log-file descriptor

number Is logid number for the logical-log file

flags Provides the status of each log as follows:

- B** Backed up
- C** Current logical-log file
- F** Free, available for use
- U** Used

uniqid Is the unique ID number of the log

begin Is the beginning page of the log file

size Is the size of the log in pages

used Is the number of pages used

%used Is the percent of pages used

active Is the number of active temporary logical logs

onstat -L command: Print the number of free locks

Use the **onstat -L** command to print the number of free locks on a lock-free list.

Syntax:

►► onstat — -L ————— ►►

Example output - TEST WITH TABS VERSION 1

Distribution of available locks on the lock free lists						
num	list head	available locks	LOCKNAME	LOCKTYPE	LISTTYPE	LISTVALUE
0	10a143b70	19996	NAME1	TYPE1	LIST1	VALUE1
1	101010101	200	NAME2	TYPE2	LIST2	VALUE2
3	020202020	300	NAME3	TYPE3	LIST3	VALUE3

Figure 20-134. **onstat -L** output

Example output - TEST WITH TABS VERSION 2 (2 TABS BETWEEN)

```

Distribution of available locks on the lock free lists

num list head  available locks LOCKNAME LOCKTYPE LISTTYPE LISTVALUE
0  10a143b70  19996 NAME1 TYPE1 LIST1 VALUE1
1  101010101  200 NAME2 TYPE2 LIST2 VALUE2
3  020202020  300 NAME3 TYPE3 LIST3 VALUE3

```

Figure 20-135. **onstat -L** output

Example output - TEST WITH SPACES - VERSION 2

```

num list head  available locks LOCKNAME LOCKTYPE LISTTYPE LISTVALUE
0  10a143b70  19996 NAME1 TYPE1 LIST1 VALUE1
1  101010101  200 NAME2 TYPE2 LIST2 VALUE2
3  020202020  300 NAME3 TYPE3 LIST3 VALUE3

```

Figure 20-136. **onstat -L** output

Example output - TEST WITH SPACES - VERSION 2

```

num list head  available locks LOCKNAME LOCKTYPE LISTTYPE LISTVALUE
0  10a143b70  19996 NAME1 TYPE1 LIST1 VALUE1
1  101010101  200 NAME2 TYPE2 LIST2 VALUE2
3  020202020  300 NAME3 TYPE3 LIST3 VALUE3

```

Figure 20-137. **onstat -L** output

Output description

num The list number

list head
The starting address of the list

available locks
The number of locks on this list

onstat -m command: Print recent system message log information

Use the **onstat -m** command to display the 20 most recent lines of the system message log.

You can use the **onstat -m** command option with the database server in any mode, including offline.

Syntax:

►► onstat — -m ◀◀

Example output

Output from this command lists the full pathname of the message log file and the 20 file entries. A date-and-time header separates the entries for each day. A time stamp prefaces single entries within each day. The name of the message log is specified as MSGPATH in the **ONCONFIG** file.

```
Message Log File: /work/11.50/dbspaces/star3.log
11:26:33 Checkpoint Completed: duration was 0 seconds.
11:26:33 Checkpoint loguniqu 1, logpos 0x23c408, timestamp: 0x2cc2 Interval: 9
```

Figure 20-138. **onstat -m** command output

onstat -o command: Output shared memory contents to a file

Use the **onstat -o** command to write the contents of shared memory to a specified file for later analysis. If you do not specify an output file, a file named **onstat.out** is created in the current directory.

Syntax:

```
▶▶ onstat -o [nobuffs] [full] outfile ▶▶
```

Use the **nobuffs** option to exclude the buffer pool in the resident segment of shared memory from the output file. This results in a smaller output file.

Use the **full** option to create an output file that is the same size as the shared memory segments for the IBM Informix instance. You must have enough room in the file system to handle the output.

If you do not specify either the **nobuffs** or the **full** option, the output is controlled by the database server DUMPSHMEM configuration parameter setting:

- If DUMPSHMEM is set to 0 or to 1, **onstat -o** command writes a full shared-memory dump file.
- If DUMPSHMEM is set to 2, **onstat -o** command writes a **nobuffs** shared-memory dump file that excludes the buffer pool in the resident segment.

By running additional **onstat** commands against the file, you can gather information from a previously saved shared memory dump. The *outfile* that you create with the **onstat -o** command is the *infile* that you can use as a source file to run the additional **onstat** commands. For more information, see “Running **onstat** Commands on a Shared Memory Dump File” on page 20-25.

onstat -O command: Print optical subsystem information

Use the **onstat -O** command to display information about the Optical Subsystem memory cache and the staging area blobspace.

Syntax:

```
▶▶ onstat -O ▶▶
```

Example output

The totals shown in the display accumulate from session to session. The database server resets the totals to 0 only when you run the **onstat -z** command.

Optical StageBlob Cache							
System Cache Totals:				System Blob Totals:			
Size	Alloc.	Avail.	Number	Kbytes	Number	Kbytes	
500	500	0	1	20	3	1500	
User Cache Totals:				User Blob Totals:			
SID	User	Size	Number	Kbytes	Number	Kbytes	
94	doug	250	1	20	1	300	
95	beth	500	0	0	2	1200	

Figure 20-139. **onstat -O** command output

Output description

The first section of the display provides the following information on system-cache totals:

size Is the size that the OPCACHEMAX configuration parameter specifies

alloc Is the number of 1-kilobyte allocations to the cache

avail Describes how much of **alloc** (in kilobytes) is not used

number Is the number of simple large objects that the database server successfully put in the cache without overflowing

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server put in the cache without overflowing

number Is the number of simple large objects that the database server wrote to the staging-area blobspace

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server wrote to the staging-area blobspace

Although the **size** output indicates the amount of memory that is specified in the configuration parameter OPCACHEMAX, the database server does not allocate memory to OPCACHEMAX until necessary. Therefore, the **alloc** output reflects only the number of 1-kilobyte allocations of the largest simple large object that has been processed. When the values in the **alloc** and **avail** output are equal to each other, the cache is empty.

The second section of the display describes the following user-cache totals information:

SID Is the session ID for the user

user Is the user ID of the client

size Is the size specified in the **INFORMIXOPCACHE** environment variable, if it is set

If you do not set the **INFORMIXOPCACHE** environment variable, the database server uses the size that you specify in the configuration parameter OPCACHEMAX.

number Is the number of simple large objects that the database server put into cache without overflowing

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server put in the cache without overflowing

number

Is the number of simple large objects that the database server wrote to the staging-area blobspace

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server wrote to the staging-area blobspace

The last line of the display lists the total number of sessions that are using the cache.

onstat -p command: Print profile counts

Use the **onstat -p** command to display information about profile counts either since you started the database server or since you ran the **onstat -z** command.

Syntax:

►► onstat — -p ————— ►►

Example output

```
Profile
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
939      943      143905  99.35  3925      10816   46919  91.63

isamtot  open    start   read    write   rewrite  delete   commit  rollbk
100055   15851   16112   24632   13343   1342     1392     905     0

gp_read  gp_write gp_rewrt gp_del   gp_alloc gp_free   gp_curs
0         0         0         0         0         0         0

ovlock   ovuserth ovbuff   usercpu  syscpu   numckpts flushes
0         0         0         12.00    2.69     9         101

bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
8         0         26894    0         0         1         1247    478

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
5         0         10       15         23
```

Figure 20-140. **onstat -p** command output

Output description

The first portion of the output describes reads and writes.

Reads and writes are tabulated in three categories: from disk, from buffers, and number of pages (read or written).

The first **%cached** field is a measure of the number of reads from buffers compared to reads from disk. The second **%cached** field is a measure of the number of writes to buffers compared to writes to disk.

The database server buffers the information and writes the information to the disk in pages. For this reason, the number of disk writes displayed as `dskwrits` is usually less than the number of writes that an individual user runs:

dskreads

The number of actual reads from disk

pagreads

The number of pages read

bufreads

Is the number of reads from shared memory

%cached

The percent of reads cached in the buffer pool.

If `bufreads` exceeds the maximum integer (or long) value, its internal representation becomes a negative number, but the value appears as 0.0.

dskwrits

The actual number of physical writes to disk

This number includes the writes for the physical and logical logs reported in **onstat -l**.

pagwrits

The number of pages written

bufwrits

The number of writes to shared memory

%cached

The percent of writes cached in the buffer pool.

The next portion of the **-p** display tabulates the number of times different ISAM calls were executed. The calls occur at the lowest level of operation and do not necessarily correspond one-to-one with SQL statement execution. A single query might generate multiple ISAM calls. These statistics are gathered across the database server and cannot be used to monitor activity on a single database unless only one database is active or only one database exists:

isamtot The total number of calls

open Increments when a `tblspace` is opened

start Increments the pointer within an index

read Increments when the read function is called

write Increments with each write call

rewrite Increments when an update occurs

delete Increments when a row is deleted

commit Increments each time that an **iscommit()** call is made

No one-to-one correspondence exists between this value and the number of explicit COMMIT WORK statements that are executed.

rollbk Increments when a transaction is rolled back

The next portion of the **onstat -p** command output displays information about generic pages. The Generic Page Manager provides an API for Informix to manage nonstandard pages in the database server buffer pool. The following table describes the Generic Page Manager fields in the **onstat -p** command output.

gp_read The number of generic page reads

gp_write The number of generic page writes

gp_rewrt The number of generic page updates

gp_del The number of generic page deletes

gp_alloc The number of generic page allocations

gp_free The number of generic pages freed and returned to tblspaces

gp_curs The number of cursors used against generic pages

The next portion of the **onstat -p** command output displays the number of times that a resource was requested when none was available:

ovlock The number of times that the database server attempted to allocate locks more than 15 times

For more information, see “LOCKS” on page 1-56.

ovuserthread The number of times that a user attempted to exceed the maximum number of user threads

ovbuff The number of times that the database server could not find a free shared-memory buffer

When no buffers are free, the database server writes a dirty buffer to disk and then tries to find a free buffer.

usercpu Is the total user CPU time that all user threads use, expressed in seconds
This entry is updated every 15 seconds.

syscpu The total system CPU time that all user threads use, expressed in seconds
This entry is updated every 15 seconds.

numckpts The number of checkpoints since the boot time

flushes The number of times that the buffer pool has been flushed to the disk

The next portion of the **onstat -p** command output contains miscellaneous information, as follows:

bufwaits Increments each time that a user thread must wait for a buffer

lokwaits Increments each time that a user thread must wait for a lock

lockreqs Increments each time that a lock is requested

deadlks Increments each time that a potential deadlock is detected and prevented

dltouts Increments each time that the distributed deadlock time-out value is exceeded while a user thread is waiting for a lock

ckpwaits

Is the number of checkpoint waits

compress

Increments each time that a data page is compressed

seqscans

Increments for each sequential scan

The last portion of the **onstat -p** command output contains the following information:

ixda-RA

The count of read-aheads that go from index leaves to data pages

idx-RA The count of read-aheads that traverse index leaves

da-RA The count of data-path-only scans

RA-pgsused

Indicates the number of pages used that the database server read ahead

lchwaits

Stores the number of times that a thread was required to wait for a shared-memory latch

A large number of latch waits typically results from a high volume of processing activity in which the database server is logging most of the transactions.

onstat -P command: Print partition information

Use the **onstat -P** command to display the partition number and the pages in the buffer pool for all of the partitions.

Syntax:

►► onstat — -P ————— ►◀

For information about running **onstat -P** on a dump file created without the buffer pool, see “Running **onstat** Commands on a Shared Memory Dump File” on page 20-25.

Example output

```

Buffer pool page size: 2048
partnum  total    btree    data    other    dirty
0         36      1        8       27       0
1048577   2        0        0        2       0
1048578   4        1        1        2       0
1048579   23       10       12       1       0
1048580   68       31       36       1       0
4194309   3         0        1        2       0

Totals:  3000      786      1779      435      0
Percentages:
Data  59.30
Btree 26.20
Other 14.50

Buffer pool page size: 8192
partnum  total    btree    data    other    dirty
0         999      0        0       999      0
5242881   1         0        0        1       0

Totals:  1000      0        0       1000      0
Percentages:
Data  0.00
Btree 0.00
Other 100.00

```

Figure 20-141. **onstat -P** command output

Output description

Buffer pool page size

The size, in bytes, of the buffer pool pages.

partnum

The partition number.

total The total number of partitions.

btree The number of B-tree pages in the partition.

data The number of data pages in the partition.

other The number of other pages in the partition.

dirty The number of dirty pages in the partition.

onstat -r command: Repeatedly print selected statistics

Use the **onstat -r** command to repeatedly print the statistics for other options specified in the command at specified intervals.

Syntax:

```

>> onstat -r [seconds] [other_options]
               |
               +-----other_options----->>

```

Use the **onstat -r seconds other_options** command to specify the seconds to repeat the other option.

Use **onstat -r *other_options*** command to have the option repeat every five seconds, which allows the other options to be concatenated with the **-r** option, as in this example: `onstat -rFh`.

The **onstat -r** command can be used in both command mode and interactive mode, and can be useful for repeating command output to monitor system resource utilization.

Example output running the onstat -r command every five seconds

```
onstat -r
IBM Informix Dynamic Server Version 11.70.F      -- On-Line -- Up 20:05:25 -- 1067288 Kbytes
IBM Informix Dynamic Server Version 11.70.F      -- On-Line -- Up 20:05:30 -- 1067288 Kbytes
IBM Informix Dynamic Server Version 11.70.F      -- On-Line -- Up 20:05:35 -- 1067288 Kbytes
```

Figure 20-142. command output

Example output running the onstat -r command every ten seconds

```
onstat -r 10
IBM Informix Dynamic Server Version 11.50.F      -- On-Line -- Up 20:06:58 -- 1067288 Kbytes
IBM Informix Dynamic Server Version 11.50.F      -- On-Line -- Up 20:07:08 -- 1067288 Kbytes
IBM Informix Dynamic Server Version 11.50.F      -- On-Line -- Up 20:07:18 -- 1067288 Kbytes
```

Figure 20-143. command output

Example output running the onstat -r every one second, with the -h option

```

onstat -r 1 -h

Buffer pool page size: 2048

buffer hash chain length histogram
# of chains      of len
    3841          0
    3767          1
    522           2
    62            3
    8192 total chains
    4351 hashed buffs
    5000 total buffs

Buffer pool page size: 2048

buffer hash chain length histogram
# of chains      of len
    4020          0
    3392          1
    735           2
    43            3
    2             4
    8192 total chains
    4172 hashed buffs
    5000 total buffs

```

Figure 20-144. **onstat -r 1 -h** command output

Example output running the onstat -r command every five seconds, with the -Fh options

```

onstat -rFh

Fg Writes      LRU Writes      Chunk Writes
0              0              21

address        flusher  state  data  # LRU  Chunk  Wakeups  Idle Tim
460e6820       0        I      0     0      2      5        9.820
      states: Exit Idle Chunk Lru

Buffer pool page size: 2048

buffer hash chain length histogram
# of chains      of len
      6342         0
      1850         1
      8192    total chains
      1850    hashed buffs
      5000    total buffs

Fg Writes      LRU Writes      Chunk Writes
0              0              21

address        flusher  state  data  # LRU  Chunk  Wakeups  Idle Tim
460e6820       0        I      0     0      2     10     22.755
      states: Exit Idle Chunk Lru

Buffer pool page size: 2048

buffer hash chain length histogram
# of chains      of len
      4396         0
      3796         1
      8192    total chains
      3796    hashed buffs
      5000    total buffs

```

Figure 20-145. **onstat -rFh** command output

onstat -R command: Print LRU, FLRU, and MLRU queue information

Use the **onstat -R** command to display detailed information about the LRU queues, FLRU queues, and MLRU queues. For each queue, the **onstat -R** command displays the number of buffers in the queue and the number and percentage of buffers that have been modified.

For an in-depth discussion of the three types of queues, see LRU queues in the shared-memory chapter of the *IBM Informix Administrator's Guide*.

Syntax:

```

▶▶—onstat— -R—————▶▶

```

Example output

```

Buffer pool page size: 2048
8 buffer LRU queue pairs
# f/m pair total % of length LOW HIGH
0 f 375 100.0% 375 375 0
1 m 0 0.0% 0 0 0
2 f 375 100.0% 375 375 0
3 m 0 0.0% 0 0 0
4 f 375 100.0% 375 375 0
5 m 0 0.0% 0 0 0
6 F 375 100.0% 375 375 0
7 m 0 0.0% 0 0 0
8 f 375 100.0% 375 375 0
9 m 0 0.0% 0 0 0
10 f 375 100.0% 375 375 0
11 m 0 0.0% 0 0 0
12 f 375 100.0% 375 375 0
13 m 0 0.0% 0 0 0
14 f 375 100.0% 375 375 0
15 m 0 0.0% 0 0 0
0 dirty, 3000 queued, 3000 total, 4096 hash buckets, 2048 buffer size
start clean at 60.000% (of pair total) dirty, or 226 buffs dirty, stop at
50.000%
Buffer pool page size: 8192
4 buffer LRU queue pairs
# f/m pair total % of length LOW HIGH
0 F 250 100.0% 250 250 0
1 m 0 0.0% 0 0 0
2 f 250 100.0% 250 250 0
3 m 0 0.0% 0 0 0
4 f 250 100.0% 250 250 0
5 m 0 0.0% 0 0 0
6 f 250 100.0% 250 250 0
7 m 0 0.0% 0 0 0
0 dirty, 1000 queued, 1000 total, 1024 hash buckets, 8192 buffer size
start clean at 60.000% (of pair total) dirty, or 150 buffs dirty, stop at
50.000%

```

Figure 20-146. **onstat -R** command output

Output description

Buffer pool page size

Is the page size of the buffer pool in bytes

Shows the queue number

Each LRU queue is composed of two subqueues: an FLRU queue and a MLRU queue. (For a definition of FLRU and MLRU queues, see LRU queues in the shared-memory chapter of the *IBM Informix Administrator's Guide*.) Thus, queues 0 and 1 belong to the first LRU queue, queues 2 and 3 belong to the second LRU queue, and so on.

f/m Identifies queue type

This field has four possible values:

f Free LRU queue

In this context, free means not modified. Although nearly all the buffers in an LRU queue are available for use, the database server attempts to use buffers from the FLRU queue rather than the MLRU queue. (A modified buffer must be written to disk before the database server can use the buffer.)

F Free LRU with fewest elements

The database server uses this estimate to determine where to put unmodified (free) buffers next.

m MLRU queue

M MLRU queue that a flusher is cleaning

length Tracks the length of the queue measured in buffers

% of Shows the percent of LRU queue that this subqueue composes

For example, suppose that an LRU queue has 50 buffers, with 30 of those buffers in the MLRU queue and 20 in the FLRU queue. The **% of** column would list percents of 60.00 and 40.00, respectively.

pair total

Provides the total number of buffers in this LRU queue

priority levels

Displays the priority levels: LOW, MED_LOW, MED_HIGH, HIGH

The **onstat -R** command also lists the priority levels.

Summary information follows the individual LRU queue information. You can interpret the summary information as follows:

dirty Is the total number of buffers that have been modified in all LRU queues

queued Is the total number of buffers in LRU queues

total Is the total number of buffers

hash buckets

Is the number of hash buckets

buffer size

Is the size of each buffer

start clean

Is the value specified in the **lru_max_dirty** field of the BUFFERPOOL configuration parameter

stop at Is the value specified in the **lru_min_dirty** field of the BUFFERPOOL configuration parameter

priority downgrades

Is the number of LRU queues downgraded to a lower priority.

priority upgrades

Is the number of LRU queues upgraded to a higher priority.

onstat -s command: Print latch information

Use the **onstat -s** command to display general latch information, including the resource that the latch controls.

Syntax:

►► onstat — -s ————— ►►

Example output

Latches with lock or userthread set				
name	address	lock	wait	userthread

Figure 20-147. **onstat -s** command output

Output description

name Identifies the resource that the latch controls with the following abbreviations:

- archive** Storage-space backup
- bf** Buffers
- bh** Hash buffers
- chunks** Chunk table
- ckpt** Checkpoints
- dbspace** Dbspace table
- flushctl** Page-flusher control
- flushr** Page cleaners
- locks** Lock table
- loglog** Logical log
- LRU** LRU queues
- physb1** First physical-log buffer
- physb2** Second physical-log buffer
- physlog** Physical log
- pt** Tblspace tblspace
- tblsps** Tblspace table
- users** User table

address Is the address of the latch

This address appears in the **onstat -u** (users) command output wait field if a thread is waiting for the latch.

lock Indicates if the latch is locked and set

The codes that indicate the lock status (1 or 0) are computer dependent.

wait Indicates if any user thread is waiting for the latch

userthread

Is the shared-memory address of any user thread that is waiting for a latch

Instead this field contains the thread-control block address, which all threads have. You can compare this address with the user addresses in the **onstat -u** output to obtain the user-process identification number.

To obtain the **rstcb** address from the **tc**b address, examine the output of the **onstat -g ath** command, which lists both addresses for each user thread.

onstat -t and onstat -T commands: Print tblspace information

Use the **onstat -t** command to display tblspace information for active tblspaces. Use the **onstat -T** command to display tblspace information for all tblspaces.

The **onstat -t** command also lists the number of active tblspaces and the total number of tblspaces.

Syntax:



Example output

Tblspaces										
n	address	flgs	ucnt	tblnum	physaddr	npages	nused	npdata	nrows	nextns
62	a40dc70	0	1	100001	1:14	250	250	0	0	1
195	ac843e0	0	1	1000df	1:236	16	9	4	53	2
2 active, 221 total										

Figure 20-148. **onstat -t** command output

Output description

- n* Is a counter of open tblspaces
- address* Is the address of the tblspace in the shared-memory tblspace table
- flgs* Uses the following flag bits to describe the flag:
 - 0x00000001**
Partition structure is being initialized
 - 0x00000002**
Partition was modified. The modified pages have not been flushed to disk.
 - 0x00000004**
Partition is being dropped
 - 0x00000008**
Partition is for a pseudo table
 - 0x00000010**
Partition is being altered in an ADD INDEX or DROP INDEX operation
 - 0x00000020**
Partition is being altered in an ALTER TABLE operation
 - 0x00000080**
Partition is being dropped while the dbspace is down

0x00000100	Simple large objects in blobspaces are not deleted when the table is dropped
0x00000200	Partition alter page count is updated
0x00000400	Pages have been altered to the latest database schema
0x00000800	System temp table
0x00001000	User temp table
0x00004000	Index operations are deferred during recovery
0x00008000	Partition is being truncated
0x00010000	Partition is partially truncated
<i>ucnt</i>	Is the usage count, which indicates the number of user threads currently accessing the tblspace
<i>tblnum</i>	Is the tblspace number expressed as a hexadecimal value The integer equivalent appears as the partnum value in the systables system catalog table.
<i>physaddr</i>	Is the physical address (on disk) of the tblspace
<i>npages</i>	Is the number of pages allocated to the tblspace
<i>nused</i>	Is the number of used pages in the tblspace
<i>npdata</i>	Is the number of data pages used
<i>nrows</i>	Is the number of data rows used
<i>nextns</i>	Is the number of noncontiguous extents allocated This number is not the same as the number of times that a next extent has been allocated.

onstat -u command: Print user activity profile

Use the **onstat -u** command to display a profile of user activity.

Syntax:

►► onstat — -u ————— ◄◄

Example output

```

Userthreads
address  flags  sessid  user    tty    wait    tout  locks  nreads  nwrites
a4d8018  ---P--D 1      informix -      0      0      0      58      4595
a4d8628  ---P--F 0      informix -      0      0      0      0      2734
a4d8c38  ---P--- 5      informix -      0      0      0      0      1
a4d9248  ---P--B 6      informix -      0      0      0      40      0
a4d9858  ---P--D 7      informix -      0      0      0      0      0
a4d9e68  Y--P--- 21     niraj   -      a65e5a8 0      1      0      0
6 active, 128 total, 7 maximum concurrent

```

Figure 20-149. **onstat -u** command output

Output description

The **-u** option displays the following output for each user thread.

address The shared-memory address of the user thread in the user table.

Compare this address with the addresses displayed in the output from the **-s** option (latches); the output from the **-b**, **-B**, and **-X** options (buffers); and the output from the **-k** option (locks) to learn what resources this thread is holding or waiting for.

flags Provides the status of the session.

The flag codes for position 1:

- B** Waiting for a buffer
- C** Waiting for a checkpoint
- G** Waiting for a write of the logical-log buffer
- L** Waiting for a lock
- S** Waiting for mutex
- T** Waiting for a transaction
- Y** Waiting for condition
- X** Waiting for a transaction cleanup (rollback)

DEFUNCT

The thread has incurred a serious assertion failure, and has been suspended to allow other threads to continue their work.

The flag code for position 2:

- *** Transaction active during an I/O failure

The flag code for position 3:

- A** A dbspace backup thread

For other values that appear here, see the third position of flag codes for the **-x** option.

The flag code for position 4:

- P** Primary thread for a session

The flag codes for position 5:

- R** Reading
- X** Thread in critical section

The flag codes for position 6:

- R Thread used in recovery (for example, physical or logical recovery)
- Thread not used in recovery

The flag codes for position 7:

- B A B-tree cleaner thread
- C Terminated user thread waiting for cleanup
- D A daemon thread
- F A page-cleaner thread
- M Special ON-Monitor thread (UNIX)

sessid The session identification number.

During operations such as parallel sorting and parallel index building, a session might have many user threads associated with it. For this reason, the session ID identifies each unique session.

user The user login name, which is derived from the operating system

tty The name of the standard error (stderr) file that the user is using, which is derived from the operating system.

This field is blank on Windows.

wait If the user thread is waiting for a specific latch, lock, mutex, or condition, this field displays the address of the resource. Use this address to map to information provided in the output from the **-s** (latch) option or the **-k** (lock) option. If the wait is for a persistent condition, run a **grep** for the address in the output from the **onstat -a** command.

tout The number of seconds left in the current wait

If the value is 0, the user thread is not waiting for a latch or lock. If the value is -1, the user thread is in an indefinite wait.

locks The number of locks that the user thread is holding

The **-k** output should include a listing for each lock held.)

nreads The number of disk reads that the user thread has executed

nwrites

The number of write calls that the user thread has executed

All write calls are writes to the shared-memory buffer cache.

The last line of the **onstat -u** command output displays the maximum number of concurrent user threads that were allocated since you initialized the database server. For example, the last line of a sample **onstat -u** command output is as follows:

```
4 active, 128 total, 17 maximum concurrent
```

The last part of the line, 17 maximum concurrent, indicates that the maximum number of user threads that were running concurrently since you initialized the database server is 17.

The output also indicates the number of active users and the maximum number of users allowed.

onstat -x command: Print database server transaction information

Use the **onstat -x** command to display transaction information on the database server.

Syntax:

```
➤ onstat -x ➤
```

The transaction information is required only in the following situations:

- X/Open environment
- Database server participation in distributed queries
- Database server uses the Microsoft Transaction Server (MTS) transaction manager

Example output

Transactions									
address	flags	userthread	locks	begin logpos	current logpos	isol	est. rb_time	retrys	coord
a6d8028	A----	a695028	0	-	-	COMMIT	-	0	
a6d8348	A----	a695878	0	-	-	COMMIT	-	0	
a6d8668	A----	a6960c8	0	-	-	COMMIT	-	0	
a6d8988	A----	a696918	0	-	-	COMMIT	-	0	
a6d8fc8	A----	a698208	0	-	-	COMMIT	-	0	
a6d92e8	A----	a6979b8	0	-	-	COMMIT	-	0	
a6d9608	A----	a698a58	0	-	-	COMMIT	-	0	
a6d9928	A----	a6992a8	1	-	-	DIRTY	-	0	
a6d9c48	A----	a6992a8	0	-	-	NOTRANS	-	0	
a6d9f68	A----	a69a348	0	-	-	COMMIT	-	0	
a6da288	A----	a69ab98	0	-	-	COMMIT	-	0	
a6da5a8	A----	a69b3e8	0	-	-	COMMIT	-	0	
a6da8c8	A----	a69bc38	0	-	-	COMMIT	-	0	
a6dabe8	A----	a69c488	0	-	-	COMMIT	-	0	
a6daf08	A----	a699af8	0	-	-	COMMIT	-	0	
a6db228	A----	a6992a8	0	-	-	COMMIT	-	0	
a6db548	A----	a69ccd8	1	-	-	DIRTY	-	0	
a6db868	A----	a69d528	1	-	-	DIRTY	-	0	
a6dbb88	A----	a69ccd8	0	-	-	COMMIT	-	0	
a6dbea8	A----	a69dd78	0	-	-	COMMIT	-	0	
a6dc1c8	A----	a69e5c8	0	-	-	COMMIT	-	0	
a6dc4e8	A-B--	a69ee18	502	33:0x25018	34:0x486fc	COMMIT	0:07	0	
22 active, 128 total, 23 maximum concurrent									

Figure 20-150. onstat -x command output

Output description

You can interpret output from the **onstat -x** command as follows:

- address** The shared-memory address of the transaction structure
- flags** The flag codes for position 1 (current transaction state):
- A** User thread attached to the transaction
 - S** TP/XA suspended transaction

C TP/XA waiting for rollback

The flag codes for position 2 (transaction mode):

T Tightly-coupled mode (MTS)

L Loosely-coupled mode (default mode)

The flag codes for position 3 (transaction stage):

B Begin work

P Distributed query prepared for commit

X TP/XA prepared for commit

C Committing or committed

R Rolling back or rolled back

H Heuristically rolling back or rolled back

The flag code for position 4:

X XA transaction

The flag codes for position 5 (type of transaction):

G Global transaction

C Distributed query coordinator

S Distributed query subordinate

B Both distributed query coordinator and subordinate

M Redirected global transaction

userthread

The thread that owns the transaction (**rstcb** address)

locks The number of locks that the transaction holds

begin logpos

The position within the log when the BEGIN WORK record was logged.

current logpos

The current log position of the most recent record that the transaction is wrote too (As a transaction rolls back, the current log position will actually wind back until it gets to the beginning log position. When the beginning log position is reached, the rollback is complete.)

isol The isolation level.

est. rb time

The estimated time the server needs to rollback the transaction. As a transaction goes forward, this time increases. If a transaction rolls back, the time decreases as the transaction unwinds.

retrys Are the attempts to start a recovery thread for the distributed query

coord The name of the transaction coordinator when the subordinate is executing the transaction

This field tells you which database server is coordinating the two-phase commit.

The last line of the **onstat -x** command output indicates that 8 is the maximum number of concurrent transactions since you initialized the database server.

8 active, 128 total, 8 maximum concurrent

Determine the position of a logical-log record

Use the **onstat -x** command to determine the position of a logical-log record.

The **curlog** and **logposit** fields provide the exact position of a logical-log record. If a transaction is not rolling back, **curlog** and **logposit** describe the position of the most recently written log record. When a transaction is rolling back, these fields describe the position of the most recently “undone” log record. As the transaction rolls back, the **curlog** and **logposit** values decrease. In a long transaction, the rate at which the **logposit** and **beginlg** values converge can help you estimate how much longer the rollback is going to take.

For an **onstat -x** command example, see monitoring a global transaction in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*.

Determine the mode of a global transaction

The **onstat -x** command is useful for determining whether a global transaction is executing in loosely-coupled or tightly-coupled mode.

The second position of the flags column in the output from the **onstat -x** command displays the flags for global transactions. The T flag indicates tightly-coupled mode and the L flag indicates loosely-coupled mode.

- *Loosely-coupled mode* means that the different database servers coordinate transactions but do not share locks. Each branch in a global transaction has a separate transaction XID. The records from all branches display as separate transactions in the logical log.
- *Tightly-coupled mode* means that the different database servers coordinate transactions and share resources such as locking and logging. In a global transaction, all branches that access the same database share the same transaction XID. Log records for branches with the same XID appear under the same session ID. MTS uses tightly-coupled mode.

onstat -X command: Print thread information

Use the **onstat -X** command to obtain precise information about the threads that are waiting for buffers.

For each buffer in use, the **onstat -X** command displays general buffer information that is also available with either the **onstat -b** or **onstat -B** commands. For more information, refer to the **onstat -b** command in “**onstat -b** command: Print buffer information for buffers in use” on page 20-26.

Syntax:

►► onstat — -X ————— ◄◄

Example output

```

    Buffers (Access)
address owner  flags pagenum      memaddr  nslots pgflgs scout  waiter
    Buffer pool page size: 2048
    0 modified, 3000 total, 4096 hash buckets, 2048 buffer size
    Buffer pool page size: 8192
    0 modified, 1000 total, 1024 hash buckets, 8192 buffer size

```

Figure 20-151. **onstat -X** command output

Output description

The **onstat -X** command has a **waiter** field to list all user threads that are waiting for the buffer, whereas the **onstat -b** and **onstat -B** commands contain a **waitlist** field that displays the address of the first user thread that is waiting for the buffer. The maximum number of shared buffers is specified in the **buffers** field in the BUFFERPOOL configuration parameter in the ONCONFIG file.

Buffer pool page size

The size of the buffer pool pages in bytes

address

The address of the buffer header in the buffer table

flags Flags identifying the current status of the buffer page:

0x01 Modified Data
0x02 Data
0x04 LRU
0x08 Error
0x20 LRU AIO write in progress
0x40 Chunk write in progress
0x80 Buffer is/will be result of read-ahead
0x100 Cleaner assigned to LRU
0x200 Buffer should avoid bf_check calls
0x400 Do log flush before writing page
0x800 Buffer has been 'buff' -checked
0x8000 Buffer has been pinned

pagenum

The physical page number on the disk

memaddr

The buffer memory address

nslots The number of slot-table entries in the page

This field indicates the number of rows (or portions of a row) that are stored on the page.

pgflgs Uses the following values, alone or in combination, to describe the page type:

1 Data page
2 Tblspace page
4 Free-list page
8 Chunk free-list page

9	Remainder data page
b	Partition resident blobpage
c	Blobspace resident blobpage
d	Blob chunk free-list bit page
e	Blob chunk blob map page
10	B-tree node page
20	B-tree root-node page
40	B-tree branch-node page
80	B-tree leaf-node page
100	Logical-log page
200	Last page of logical log
400	Sync page of logical log
800	Physical log
1000	Reserved root page
2000	No physical log required
8000	B-tree leaf with default flags

scount Displays the number of threads that are waiting for the buffer

waiter Lists the addresses of all user threads that are waiting for the buffer

onstat -z command: Clear statistics

Use the **onstat -z** command to clear database server statistics, including statistics that relate to Enterprise Replication, and set the profile counts to 0.

If you use the **onstat -z** command to reset and monitor the count of some fields, be aware that profile counts are incremented for all activity that occurs in any database that the database server manages. Any user can reset the profile counts and thus interfere with monitoring that another user is conducting.

Syntax:

►► onstat — -z ————— ►►

Return codes on exiting the onstat utility

The **onstat** utility displays a set of return codes when you exit the utility.

Example

The following lines are an example of the messages and return codes that are displayed when you exit the **onstat** utility:

```
GLS failures: -1
Failed to attach shared memory: -1
Failed to attach shared memory when running 'onstat -': 255
All other errors detected by onstat: 1
No errors detected by onstat: 0
Administration mode: 7
```

Return code values

The following table lists the database server mode that corresponds to the return codes that are displayed when you exit the **onstat** utility.

Value	Explanation
-1	GLS locale initialization failed or Informix failed to attach to shared memory
0	Initialization mode
1	Quiescent mode
2	Recovery mode
3	Backup mode
4	Shutdown mode
5	Online mode
6	Abort mode
7	User mode
255	Off-Line mode

Part 3. SQL Administration API

Chapter 21. SQL Administration API Functions

These topics describe the SQL administration API **admin()** and **task()** functions.

SQL Administration API Overview

Use the SQL administration API to remotely administer Informix through SQL statements.

The SQL administration API consists of two functions: **admin()** and **task()**. These functions perform the same operations, but return results in different formats. These functions take one or more arguments that define the operation. Many of the operations are ones that you can also complete with command line utilities. The advantage of using the SQL administration API functions is that you can run them remotely from other database servers; whereas you must be directly connected to the database server on which to run command line utility commands.

You can invoke the **admin()** and **task()** functions within SQL statements that can include an expression, or you can use the EXECUTE FUNCTION statement to call them.

The SQL administration API functions are defined in the **sysadmin** database. You must be connected to the **sysadmin** database, either directly or remotely, to run these functions.

The SQL administration API functions can be run only by the following users:

- The user **informix**
- The **root** user, if Connect privilege on the **sysadmin** database is granted to the user
- The DBSA group members, if Connect privilege on the **sysadmin** database is granted to the role

You can generate SQL administration API commands for reproducing the storage spaces, chunks, and logs that exist in a file. To do this, run the **dbschema** utility with the **-c** option.

admin() and task() Function Syntax Behavior

The **admin()** and **task()** functions take one or more arguments as quoted strings separated by commas.

The syntax for the **admin()** and **task()** functions includes the following rules:

- Each argument must be delimited by a pair of single (') quotation marks or double (") quotation marks.
- Arguments must be separated by a comma.
- The maximum number of arguments is 28.
- Most arguments are not case-sensitive, with the following exceptions:
 - The argument that immediately follows the initial **onmode** argument is case-sensitive.

For example:

```
EXECUTE FUNCTION task("onmode","D","50");
```

- The arguments included with the **cdr** argument are case-sensitive.

For example:

```
EXECUTE FUNCTION task("cdr define server",
    "-C=g_amsterdam","--init g_amsterdam");
```

- If you are not directly connected to the **sysadmin** database, you must include the **sysadmin** database name and the server name, according to the standard Database Object Name syntax. For example, if your server name is **ids_server**, you could run the following statement:

```
EXECUTE FUNCTION sysadmin@ids_server:admin("add bufferpool", "2",
    "50000", "8", "60.0", "50.0");
```

For more information on the Database Object Name syntax, see the *IBM Informix Guide to SQL: Syntax*.

admin() and task() Argument Size Specifications

By default, the units for arguments specifying sizes in **admin()** and **task()** functions are kilobytes. You can specify other units.

You can use the following units in size arguments to the **admin()** and **task()** functions:

Notation

Corresponding Units

B	Bytes
K	Kilobytes (Default)
M	Megabytes
G	Gigabytes
T	Terabytes
P	Petabytes

The letter case of these characters is ignored.

Any white space that separates the size specification and the units abbreviation in the same argument is ignored. For example, the specifications "128M" and "128 m" are both interpreted as 128 megabytes.

When a size argument is omitted, the default size for that object applies, based either on the setting of a configuration parameter, or on the system default if no parameter is set. Storage spaces, for example, have a default size of 100 megabytes.

admin() and task() Function Return Codes

The **admin()** and **task()** functions perform equivalent tasks but produce different types of return codes. Use the **admin()** function if you want an integer return code, or the **task()** function if you want a textual return code.

When you run the **admin()** or **task()** function, it:

- Performs the specified operation.
- Returns a value that signifies whether the function succeeded or failed.
- Inserts a row into the **command_history** table of the **sysadmin** database.

The return codes for the **admin()** and **task()** functions indicate whether the function succeeded or failed in different formats:

- The **task()** function returns a textual message. The message is also inserted into the **cmd_ret_msg** column in the new row that the **task()** function inserts into the **command_history** table.
- The **admin()** function returns an integer. This number is also inserted into the **cmd_number** column in the new row that the **admin()** function inserts into the **command_history** table.
 - If this value is greater than zero, the function succeeded, and a new row was inserted into the **command_history** table.
 - If this value is zero, the function succeeded, but Informix could not insert a new row into the **command_history** table.
 - If this value is less than zero, the function failed, but a new row was inserted into the **command_history** table.

The operation that the **admin()** or **task()** function specifies occurs in a separate transaction from the insertion of the new row into the **command_history** table. If the command executes successfully, but the insertion into the **command_history** table fails, the command takes effect, but an **online.log** error entry indicates the problem.

If the **command_history.cmd_number** serial counter is 200 when this function is called, and the command succeeds, then Informix executes the command and returns the integer 201. If the command fails, this example returns the value -201.

Suppose the **task()** function had executed the same command:

```
EXECUTE FUNCTION task("check extents");
```

This command instructs the database server to check the extents, and returns a message indicating whether the command succeeded or failed.

If the **command_history.cmd_number** serial counter is 201 when this function is called, and the command fails, then the returned value is -202. Suppose that the next SQL administration API function that the DBSA invokes is this:

```
EXECUTE FUNCTION admin('create dbspace',  
    'dbspace2', '/work/CHUNKS/dbspace2', "20M");
```

If in this case the command succeeds, the returned value is 203. The DBSA can use the following query to examine the two rows of the **command_history** table that these calls to the **admin()** function inserted:

```
SELECT * FROM command_history WHERE cmd_number IN (202,203);
```

This query returns two rows:

cmd_number	202
cmd_exec_time	2009-04-17 16:26:14
cmd_user	informix
cmd_hostname	olympia
cmd_executed	create dbspace
cmd_ret_status	-1
cmd_ret_msg	Unable to create file /work/dbspace2
cmd_number	203
cmd_exec_time	2009-04-17 16:26:15
cmd_user	informix
cmd_hostname	olympia

```
cmd_executed      create dbspace
cmd_ret_status    0
cmd_ret_msg       created dbspace number 2 named dbspace2
```

SQL administration API portal: Arguments by functional category

You can view a list of **admin()** and **task()** function arguments, sorted by category, with links to information about the arguments.

Category list

To use this section, you first determine the appropriate category from the following list, then follow the link to the SQL administration API function arguments for that category.

- “Compression arguments”
- “Configuration parameter arguments” on page 21-5
- “Data, partition, and extent arguments” on page 21-6
- “Database arguments” on page 21-5
- “Enterprise replication arguments” on page 21-6
- “High availability arguments” on page 21-6
- “Listen thread arguments” on page 21-7
- “Log arguments” on page 21-8
- “Message log arguments” on page 21-8
- “Memory arguments” on page 21-8
- “Mirror arguments” on page 21-9
- “Parallel database query (PDQ) arguments” on page 21-9
- “Server mode arguments” on page 21-9
- “Space arguments” on page 21-10
- “Storage provisioning arguments” on page 21-11
- “SQL tracing arguments” on page 21-13
- “Miscellaneous arguments” on page 21-13

Backup arguments

Use the following SQL administration API function arguments to backup your databases.

Table 21-1. admin() and task() Function Arguments for Databases

Argument	When Added to Informix
“ontape archive argument: Backup the data on your database (SQL administration API)” on page 21-95	Version 11.70.xC2
“onbar argument: Backup the storage spaces (SQL administration API)” on page 21-92	Version 11.70.xC2
“onsmsync argument: Synchronize with the storage manager catalog (SQL administration API)” on page 21-93	Version 11.70.xC2

Compression arguments

Use the following SQL administration API function arguments to manage the compression of data and to optimize storage.

*Table 21-2. **admin()** and **task()** Function Arguments for Compression Commands*

Argument	When Added to Informix
“table or fragment arguments: Compress data and optimize storage (SQL administration API)” on page 21-126	Version 11.50xC4
“table or fragment arguments: Compress data and optimize storage (SQL administration API)” on page 21-126	Version 11.50xC4
“purge compression dictionary arguments: Remove compression dictionaries (SQL administration API)” on page 21-132	Version 11.50xC4

For an overview of compression and storage optimization commands, see “Compress and Uncompress Operations (SQL administration API)” on page 21-125.

Configuration parameter arguments

Use the following SQL administration API function arguments to update configuration parameters.

*Table 21-3. **admin()** and **task()** Function Arguments for Configuration Parameter Commands*

Argument	When Added to Informix
“onmode and wf arguments: Permanently update a configuration parameter (SQL administration API)” on page 21-87	Version 11.10.xC1
“onmode and wm arguments: Temporarily update a configuration parameter (SQL administration API)” on page 21-88	Version 11.10.xC1
“onmode, wm, and AUTO_LRU_TUNING arguments: Change LRU tuning status (SQL administration API)” on page 21-89	Version 11.10.xC1
“set onconfig memory argument: Temporarily change a configuration parameter (SQL administration API)” on page 21-107	Version 11.50.xC3
“set onconfig permanent argument: Permanently change a configuration parameter (SQL administration API)” on page 21-108	Version 11.50.xC3

Database arguments

Use the following SQL administration API function arguments to create and drop databases.

*Table 21-4. **admin()** and **task()** Function Arguments for Databases*

Argument	When Added to Informix
“create database argument: Create a database (SQL administration API)” on page 21-31	Version 11.70.xC2
“drop database argument: Drop a database (SQL administration API)” on page 21-46	Version 11.70.xC2

Data, partition, and extent arguments

Use the following SQL administration API function arguments to manage data, partitions, and extents.

*Table 21-5. **admin()** and **task()** Function Arguments for Data, Partition, and Extent Commands*

Argument	When Added to Informix
"onmode and C arguments: Control the B-tree scanner (SQL administration API)" on page 21-73	Version 11.10.xC1
"onmode and c arguments: Force a checkpoint (SQL administration API)" on page 21-72	Version 11.10.xC1
"check data argument: Check data consistency (SQL administration API)" on page 21-23	Version 11.10.xC1
"check extents argument: Check extent consistency (SQL administration API)" on page 21-24	Version 11.10.xC1
"check partition argument: Check partition consistency (SQL administration API)" on page 21-24	Version 11.10.xC1
"checkpoint argument: Force a checkpoint (SQL administration API)" on page 21-25	Version 11.10.xC1
"print partition argument: Print partition information (SQL administration API)" on page 21-97	Version 11.10.xC1
"set dataskip argument: Start or stop skipping a dbspace (SQL administration API)" on page 21-106	Version 11.10.xC1
"set index compression argument: Change index page compression (SQL administration API)" on page 21-106	Version 11.50.xC2

Enterprise replication arguments

Use the following SQL administration API function arguments to manage Enterprise Replication.

*Table 21-6. **admin()** and **task()** Function Arguments for Enterprise Replication Commands*

Argument	When Added to Informix
"cdr argument: Administer Enterprise Replication (SQL administration API)" on page 21-22	Version 11.50.xC3

High availability arguments

Use the following SQL administration API function arguments to manage high-availability replication (HDR).

*Table 21-7. **admin()** and **task()** Function Arguments for HDR Commands*

Argument	When Added to Informix
"archive fake argument: Perform an unrecorded backup (SQL administration API)" on page 21-21	Version 11.50.xC1
"ha make primary argument: Change the mode of a secondary server (SQL administration API)" on page 21-52	Version 11.50.xC1
"ha rss argument: Create an RS secondary server (SQL administration API)" on page 21-53	Version 11.50.xC1

*Table 21-7. **admin()** and **task()** Function Arguments for HDR Commands (continued)*

Argument	When Added to Informix
"ha rss add argument: Add an RS secondary server to a primary server (SQL administration API)" on page 21-54	Version 11.50.xC1
"ha rss change argument: Change the password of an RS secondary server (SQL administration API)" on page 21-55	Version 11.50.xC1
"ha rss delete argument: Delete an RS secondary server (SQL administration API)" on page 21-56	Version 11.50.xC1
"ha sds clear argument: Stop shared-disk replication (SQL administration API)" on page 21-56	Version 11.50.xC1
"ha sds primary argument: Convert an SD secondary server to a primary server (SQL administration API)" on page 21-57	Version 11.50.xC1
"ha sds set argument: Create a shared-disk primary server (SQL administration API)" on page 21-58	Version 11.50.xC1
"ha set idxauto argument: Replicate indexes to secondary servers (SQL administration API)" on page 21-59	Version 11.50.xC1
"ha set ipl argument: Log index builds on the primary server (SQL administration API)" on page 21-59	Version 11.50.xC1
"ha set primary argument: Define an HDR primary server (SQL administration API)" on page 21-60	Version 11.50.xC1
"ha set secondary argument: Define an HDR secondary server (SQL administration API)" on page 21-61	Version 11.50.xC1
"ha set standard argument: Convert an HDR server into a standard server (SQL administration API)" on page 21-61	Version 11.50.xC1
"ha set timeout argument: Change SD secondary server timeout (SQL administration API)" on page 21-62	Version 11.50.xC1
"onmode and d arguments: Set data-replication types (SQL administration API)" on page 21-74	Version 11.50.xC1

Listen thread arguments

Use the following SQL administration API function arguments to control listen threads for a SOCTCP or TLITCP network protocol without interrupting existing connections.

*Table 21-8. **admin()** and **task()** Function Arguments for Listen Thread Commands*

Argument	When Added to Informix
"restart listen argument: Stop and start a listen thread dynamically (SQL administration API)" on page 21-100	Version 11.50.xC6
"start listen argument: Start a listen thread dynamically (SQL administration API)" on page 21-116	Version 11.50.xC6
"stop listen argument: Stop a listen thread dynamically (SQL administration API)" on page 21-117	Version 11.50.xC6

Log arguments

Use the following SQL administration API function arguments to manage logical and physical logs.

*Table 21-9. **admin()** and **task()** Function Arguments for Log Commands*

Argument	When Added to Informix
"add log argument: Add a new logical log (SQL administration API)" on page 21-16	Version 11.10.xC1
"alter logmode argument: Change the database logging mode (SQL administration API)" on page 21-20	Version 11.10.xC1
"alter plog argument: Change the physical log (SQL administration API)" on page 21-21	Version 11.10.xC1
"drop log argument: Drop a logical log (SQL administration API)" on page 21-47	Version 11.10.xC1
"onmode and l arguments: Switch to the next logical log (SQL administration API)" on page 21-79	Version 11.10.xC1

Message log arguments

Use the following SQL administration API function arguments to manage message logs.

*Table 21-10. **admin()** and **task()** Function Arguments for Message Log Commands*

Argument	When Added to Informix
"file status argument: Display the status of a message log file (SQL administration API)" on page 21-51	Version 11.10.xC3
"message log rotate argument: Rotate the message log file (SQL administration API)" on page 21-63	Version 11.10.xC3
"message log delete argument: Delete a message log file (SQL administration API)" on page 21-63	Version 11.10.xC3
"message log truncate argument: Delete the contents of a message log file (SQL administration API)" on page 21-64	Version 11.10.xC3

Memory arguments

Use the following SQL administration API function arguments to manage memory.

*Table 21-11. **admin()** and **task()** Function Arguments for Memory Commands*

Argument	When Added to Informix
"add bufferpool argument: Add a buffer pool (SQL administration API)" on page 21-14	Version 11.10.xC1
"add memory argument: Increase shared memory (SQL administration API)" on page 21-17	Version 11.10.xC1
"onmode and a arguments: Add a shared-memory segment (SQL administration API)" on page 21-71	Version 11.10.xC1
"onmode and F arguments: Free unused memory segments (SQL administration API)" on page 21-77	Version 11.10.xC1
"onmode and n arguments: Unlock resident memory (SQL administration API)" on page 21-81	Version 11.10.xC1

Table 21-11. **admin()** and **task()** Function Arguments for Memory Commands (continued)

Argument	When Added to Informix
“onmode and r arguments: Force residency of shared memory (SQL administration API)” on page 21-84	Version 11.10.xC1
“scheduler Imm enable argument: Specify automatic low memory management settings (SQL administration API)” on page 21-101	Version 11.70.xC3
“scheduler Imm disable argument: Stop automatic low memory management (SQL administration API)” on page 21-104	Version 11.70.xC3

Mirror arguments

Use the following SQL administration API function arguments to manage mirroring.

Table 21-12. **admin()** and **task()** Function Arguments for Mirror Commands

Argument	When Added to Informix
“add mirror argument: Add a mirror chunk (SQL administration API)” on page 21-18	Version 11.10.xC1
“start mirroring argument: Starts storage space mirroring (SQL administration API)” on page 21-117	Version 11.10.xC1
“stop mirroring argument: Stops storage space mirroring (SQL administration API)” on page 21-118	Version 11.10.xC1

Parallel database query (PDQ) arguments

Use the following SQL administration API function arguments to manage PDQ.

Table 21-13. **admin()** and **task()** Function Arguments for PDQ Commands

Argument	When Added to Informix
“onmode and D arguments: Set PDQ priority (SQL administration API)” on page 21-76	Version 11.10.xC1
“onmode and M arguments: Temporarily change decision-support memory (SQL administration API)” on page 21-80	Version 11.10.xC1
“onmode and Q arguments: Set maximum number for decision-support queries (SQL administration API)” on page 21-83	Version 11.10.xC1
“onmode and S arguments: Set maximum number of decision-support scans (SQL administration API)” on page 21-85	Version 11.10.xC1

Server mode arguments

Use the following SQL administration API function arguments to change the server mode.

Table 21-14. **admin()** and **task()** Function Arguments for Server Mode Commands

Argument	When Added to Informix
"onmode and j arguments: Switch the database server to administration mode (SQL administration API)" on page 21-78	Version 11.10.xC1
"onmode and m arguments: Switch to multi-user mode (SQL administration API)" on page 21-79	Version 11.10.xC1

Space arguments

Use the following SQL administration API function arguments to manage chunks, blobspaces, dbspaces, and sbspaces.

Table 21-15. **admin()** and **task()** Function Arguments for Space Commands

Argument	When Added to Informix
"add chunk argument: Add a new chunk (SQL administration API)" on page 21-15	Version 11.10.xC1
"alter chunk argument: Change chunk status to online or offline (SQL administration API)" on page 21-19	Version 11.10.xC1
"clean sbspace argument: Release unreferenced smart large objects (SQL administration API)" on page 21-26	Version 11.10.xC1
"create blobspace argument: Create a blobspace (SQL administration API)" on page 21-26	Version 11.10.xC1
"create chunk argument: Create a chunk (SQL administration API)" on page 21-29	Version 11.10.xC1
"create dbspace argument: Create a dbspace (SQL administration API)" on page 21-32	Version 11.10.xC1
"create sbspace argument: Create an sbspace (SQL administration API)" on page 21-34	Version 11.10.xC1
"create sbspace with accesstime argument: Create an sbspace that tracks access time (SQL administration API)" on page 21-36	Version 11.70xC4
"create sbspace with log argument: Create an sbspace with transaction logging (SQL administration API)" on page 21-37	Version 11.70xC4
"create tempdbspace argument: Create a temporary dbspace (SQL administration API)" on page 21-38	Version 11.10.xC1
"create tempsbspace argument: Create a temporary sbspace (SQL administration API)" on page 21-39	Version 11.70xC4
"drop blobspace argument: Drop a blobspace (SQL administration API)" on page 21-43	Version 11.10.xC1
"drop chunk argument: Drop a chunk (SQL administration API)" on page 21-44	Version 11.10.xC1
"drop dbspace argument: Drop a dbspace (SQL administration API)" on page 21-46	Version 11.10.xC1
"drop sbspace argument: Drop an sbspace (SQL administration API)" on page 21-49	Version 11.10.xC1
"drop tempdbspace argument: Drop a temporary dbspace (SQL administration API)" on page 21-50	Version 11.10.xC1

Table 21-15. **admin()** and **task()** Function Arguments for Space Commands (continued)

Argument	When Added to Informix
"onmode and O arguments: Mark a disabled dbspace as down (SQL administration API)" on page 21-81	Version 11.10.xC1
"rename space argument: Rename a storage space (SQL administration API)" on page 21-98	Version 11.10.xC1
"set chunk argument: Change the status of a chunk (SQL administration API)" on page 21-105	Version 11.10.xC1
"set sbospace accesstime argument: Control access time tracking (SQL administration API)" on page 21-109	Version 11.10.xC1
"set sbospace avg_lo_size argument: Set the average size of smart large objects (SQL administration API)" on page 21-110	Version 11.10.xC1
"set sbospace logging argument: Change the logging of an sbospace (SQL administration API)" on page 21-111	Version 11.10.xC1

Storage provisioning arguments

Also use the following SQL administration API function arguments to manage chunks, blobspaces, dbspaces, and sbspaces.

Table 21-16. **admin()** and **task()** Function Arguments for Storage Provisioning Space Commands

Argument	When Added to Informix
"create blobspace from storagepool argument: Create a blobspace from the storage pool (SQL administration API)" on page 21-28	Version 11.70.xC1
"create chunk from storagepool argument: Create a chunk from the storage pool (SQL administration API)" on page 21-30	Version 11.70.xC1
"create dbspace from storagepool argument: Create a dbspace from the storage pool (SQL administration API)" on page 21-33	Version 11.70.xC1
"create sbospace from storagepool argument: Create an sbospace from the storage pool (SQL administration API)" on page 21-35	Version 11.70.xC1
"create tempdbspace argument: Create a temporary dbspace (SQL administration API)" on page 21-38	Version 11.70.xC1
"create tempsbospace from storagepool argument: Create a temporary sbospace from the storage pool (SQL administration API)" on page 21-41	Version 11.10.xC1
"create tempdbspace from storagepool argument: Create a temporary dbspace from the storage pool (SQL administration API)" on page 21-39	Version 11.70.xC1
"drop blobspace to storagepool argument: Return space from an empty blobspace to the storage pool (SQL administration API)" on page 21-44	Version 11.70.xC1
"drop chunk to storagepool argument: Return space from an empty chunk to the storage pool (SQL administration API)" on page 21-45	Version 11.70.xC1

*Table 21-16. **admin()** and **task()** Function Arguments for Storage Provisioning Space Commands (continued)*

Argument	When Added to Informix
“drop dbspace to storagepool argument: Return space from an empty dbspace to the storage pool (SQL administration API)” on page 21-47	Version 11.70.xC1
“drop sbspace to storagepool argument: Return space from an empty sbspace to the storage pool (SQL administration API)” on page 21-49	Version 11.70.xC1
“drop tempdbspace to storagepool argument: Return space from an empty temporary dbspace to the storage pool (SQL administration API)” on page 21-50	Version 11.70.xC1
“drop tempsbspace to storagepool argument: Return space from an empty temporary sbspace to the storage pool (SQL administration API)” on page 21-51	Version 11.70.xC1
“modify chunk extend argument: Extend the size of a chunk (SQL administration API)” on page 21-65	Version 11.70.xC1
“modify chunk extendable off argument: Mark a chunk as not extendable (SQL administration API)” on page 21-67	Version 11.70.xC1
“modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)” on page 21-66	Version 11.70.xC1
“modify space expand argument: Expand the size of a space (SQL administration API)” on page 21-68	Version 11.70.xC1
“modify space sp_sizes argument: Modify the create or extend size of a storage space (SQL administration API)” on page 21-69	Version 11.70.xC1
“storagepool add argument: Add a storage pool entry (SQL administration API)” on page 21-119	Version 11.70.xC1
“storagepool modify argument: Modify a storage pool entry (SQL administration API)” on page 21-122	Version 11.70.xC1
“storagepool delete argument: Delete one storage pool entry (SQL administration API)” on page 21-121	Version 11.70.xC1
“storagepool purge argument: Delete storage pool entries (SQL administration API)” on page 21-124	Version 11.70.xC1

SQL statement cache arguments

Use the following SQL administration API function arguments to manage the SQL statement cache.

*Table 21-17. **admin()** and **task()** Function Arguments for SQL Statement Cache Commands*

Argument	When Added to Informix
“onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)” on page 21-76	Version 11.10.xC1
“onmode and W arguments: Reset statement cache attributes (SQL administration API)” on page 21-85	Version 11.10.xC1

SQL tracing arguments

Use the following SQL administration API function arguments to manage SQL tracing.

*Table 21-18. **admin()** and **task()** Function Arguments for SQL Tracing Commands*

Argument	When Added to Informix
"set sql tracing argument: Set global SQL tracing (SQL administration API)" on page 21-111	Version 11.10.xC1
"set sql tracing database argument: Change database tracing (SQL administration API)" on page 21-113	Version 11.50.xC3
"set sql tracing session argument: Control tracing for a session (SQL administration API)" on page 21-114	Version 11.50.xC3
"set sql tracing user argument: Control tracing for users (SQL administration API)" on page 21-115	Version 11.10.xC1
"set sql user tracing argument: Set global SQL tracing for a user session (SQL administration API)" on page 21-115	Version 11.50.xC3

Miscellaneous arguments

Use the following SQL administration API function arguments to manage miscellaneous processes.

*Table 21-19. **admin()** and **task()** Function Arguments for Miscellaneous Commands*

Argument	When Added to Informix
"onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)" on page 21-76	Version 11.10.xC1
"onmode and p arguments: Add or remove virtual processors (SQL administration API)" on page 21-82	Version 11.10.xC1
"onmode and Y arguments: Change query plan measurements for a session (SQL administration API)" on page 21-90	Version 11.10.xC1
"onmode and z arguments: Terminate a user session (SQL administration API)" on page 21-91	Version 11.10.xC1
"onmode and Z arguments: Terminate a distributed transaction (SQL administration API)" on page 21-91	Version 11.10.xC1
"print error argument: Print an error message (SQL administration API)" on page 21-96	Version 11.10.xC1
"print file info argument: Display directory or file information (SQL administration API)" on page 21-96	Version 11.70.xC2
"reset sysadmin argument: Move the sysadmin database (SQL administration API)" on page 21-99	Version 11.10.xC1
"scheduler argument: Stop or start the scheduler (SQL administration API)" on page 21-101	Version 11.10.xC1

add bufferpool argument: Add a buffer pool (SQL administration API)

Use the **add bufferpool** argument with the **admin()** or **task()** function to create a buffer pool.

Syntax

```

>>EXECUTE FUNCTION admin(
└─task┘
>—"add bufferpool"—,—"pagesize"—,—"buffers"—
>└─,—"lrus"—
└─,—"max_dirty"—
└─,—"min_dirty"—
) —;

```

Element	Description	Key Considerations
<i>buffers</i>	The number of buffers. The default is 10000 buffers.	Each buffer is the size of the operating system page.
<i>lrus</i>	The number of LRU queues. The default is 8.	The maximum for 32-bit platforms is 128, and for 64-bit platforms is 512.
<i>max_dirty</i>	The percentage of modified pages in the LRU queues at which the queue is cleaned. If a field is specified out of the range of values, the default of 60.00 percent is set.	
<i>min_dirty</i>	The percentage of modified pages in the LRU queues at which page cleaning is not mandatory. Page cleaners might continue cleaning beyond this point under some circumstances. If a field is specified out of the range of values, the default of 80.00 percent is set.	
<i>pagesize</i>	The page size in kilobytes.	The page size must be an integral multiple of the default page size, and cannot be greater than 16 kilobytes. On Windows, the page size is always 4 kilobytes.

Usage

This function is equivalent to the **onparams -b -g** command and the **BUFFERPOOL** configuration parameter.

Example

The following example adds a bufferpool with a page size of 2 KB, 50,000 buffers, 8 LRU queues, an LRU maximum of 60 percent, and an LRU minimum of 50 percent:

```
EXECUTE FUNCTION task("add bufferpool","2","50000","8","60.0","50.0");
```

Related reference:

“onparams -b: Add a new buffer pool” on page 16-4

add chunk argument: Add a new chunk (SQL administration API)

Use the **add chunk** argument with the **admin()** or **task()** function to add a chunk to a dbspace or blobspace.

Syntax

```
►►EXECUTE FUNCTION admin task (—"add chunk"—,—"spacename"—,—"pathname"—►  
—,—"size"—  
—,—"offset"—  
—,—"mirror_path"—  
—,—"mirror_offset"—  
)—;—►
```

Element	Description	Key Considerations
<i>mirror_offset</i>	The location of the mirror chunk.	
<i>mirror_path</i>	The path to the mirror chunk.	If you are adding a chunk to a mirrored storage space, you must also add a mirror chunk.
<i>offset</i>	The location of the new chunk.	
<i>pathname</i>	The path of the added disk space.	
<i>size</i>	The amount of disk space to add in kilobytes.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>spacename</i>	The name of the dbspace, blobspace, or sbpace to which you are adding disk space.	

Usage

The size of the chunk must be equal to or greater than 1000 KB and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 TB.

This function is equivalent to the **onspaces -a** command.

Example

The following example adds a 5 MB chunk of raw disk space, at an offset of 5200 kilobytes, to a dbspace named **dbspc3**:

```
EXECUTE FUNCTION task("add chunk", "dbspc3", "\\.\e:", "5120", "5200");
```

The following example adds a 10 MB mirror chunk to a blobspace named **blobsp3** with an offset of 200 kilobytes for both the primary and mirror chunks:

```
EXECUTE FUNCTION task("add chunk", "blobsp3", "/dev/raw_dev1", "10240",  
"200", "/dev/raw_dev2", "200");
```

Related reference:

“onspaces -a: Add a chunk to a dbspace or blobspace” on page 19-2

“onspaces -a: Add a chunk to an sbospace” on page 19-3

add log argument: Add a new logical log (SQL administration API)

Use the **add log** argument with the **admin()** or **task()** function to add a logical log to a dbspace.

Syntax

```
►►EXECUTE FUNCTION admintask (—"add log"—,—"dbspace"—  
    ,—"size"—  
    ,—"count"—  
    ,—"after_current_flag"—  
    )—;◄◄
```

Element	Description	Key Considerations
<i>after_current_flag</i>	Whether to add the new log after the current log or after the last logical log (default).	Possible values are: <ul style="list-style-type: none">1 = Add the new log after the current log.0 = Add the new log after the last log.
<i>count</i>	The number of log files to create. The default is 1.	The number must not cause the total number of logical-log files to exceed 32,767.
<i>dbspace</i>	The name of the dbspace in which to insert a logical-log file.	<p>You can add a log file to a dbspace only if the database server has adequate contiguous space.</p> <p>You can add a log file during a backup.</p> <p>You cannot add a log file to a blobspace or sbospace.</p>

Element	Description	Key Considerations
<i>size</i>	The size in kilobytes of the new logical-log file. The default is the size specified by the LOGSIZE configuration parameter.	This value must be an unsigned integer greater than or equal to 200 KB. Also see “admin() and task() Argument Size Specifications” on page 21-2.

Usage

The newly added log files have a status of **A** and are immediately available for use. Use **onstat -l** to view the status of your logical-log files. It is recommended that you take a level-0 backup of the root dbspace and the dbspace that contains the log file as soon as possible after running this function.

By default, the new log file is added after the last logical log. Include 1 as the fifth argument to insert the logical-log file after the current log file.

This function resembles the **onparams -a -d** command, which can add a single logical-log file. You can add multiple logical-log files to the specified dbspace, however, with a single invocation of this function.

Example

The command in the following example adds three logical logs after the current log, each with a size of 5 MB:

```
EXECUTE FUNCTION task ("add log","logdbs","5M",3,1);
```

Related reference:

“onparams -a -d *dbspace*: Add a logical-log file” on page 16-2

add memory argument: Increase shared memory (SQL administration API)

Use the **add memory** argument with the **admin()** or **task()** function to add to the virtual portion of shared memory.

Syntax

```
►►EXECUTE FUNCTION admin (—"add memory"—,"size"—);
```

Element	Description	Key Considerations
<i>size</i>	The size, in kilobytes, of the new virtual shared-memory segment.	This value must not exceed the operating system limit for the size of shared-memory segments. Also see “admin() and task() Argument Size Specifications” on page 21-2.

Usage

This size defaults to the SHMADD configuration parameter.

This function is equivalent to the **onmode -a** command.

Example

The following example adds 500 KB of virtual shared-memory:

```
EXECUTE FUNCTION task("add memory","500");
```

Related reference:

“onmode -a: Add a shared-memory segment” on page 14-2

add mirror argument: Add a mirror chunk (SQL administration API)

Use the **add mirror** argument with the **admin()** or **task()** function to add a mirror chunk to a dbspace.

Syntax

```
►►EXECUTE FUNCTION admin task ("add mirror"—,"spacename"—,"path"—,►►  
►—"offset"—,"mirror_path"—,"mirror_offset"—) —;►►
```

Element	Description	Key Considerations
<i>mirror_path</i>	The disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that performs the mirroring.	
<i>mirror_offset</i>	The offset to reach the mirrored chunk of the newly mirrored dbspace, blobspace, or sbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>offset</i>	The offset into the disk partition or into the unbuffered device in kilobytes to reach the initial chunk of the newly mirrored dbspace, blobspace, or sbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>path</i>	The disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you want to mirror.	
<i>spacename</i>	The name of a dbspace, blobspace, or sbspace to mirror.	

Usage

This function is equivalent to the **onspaces -m** command.

Example

The following example adds a mirror chunk to a blobspace named **blobbsp3**:

```
EXECUTE FUNCTION task("add mirror","blobbsp3","/dev/raw_dev1",
"10240","/dev/raw_dev2","200");
```

Related reference:

“onspaces -m: Start mirroring” on page 19-22

alter chunk argument: Change chunk status to online or offline (SQL administration API)

Use the **alter chunk** argument with the **admin()** or **task()** function to bring a chunk online or take a chunk offline in a dbspace, blobspace, or sbospace.

Syntax

```
►►EXECUTE FUNCTION admin ("alter chunk offline"
task "alter chunk online"
►,—"space_name"—,—"path"—,—"offset"—)►;►
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the blobspace, dbspace, or sbospace.	
<i>path</i>	The disk partition or unbuffered device of the chunk.	
<i>offset</i>	The offset (in kilobytes) into the disk partition or unbuffered device to reach the chunk. The default is 0.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

The chunk must be in a mirrored pair, or a non-primary chunk within a noncritical dbspace.

Use the **alter chunk online** argument to change the chunk status to online.

Use the **alter chunk offline** argument to change the chunk status to offline.

This function is equivalent to the **onspaces -s** command.

Example

The following example brings a chunk in a space named **dbspace4** online:

```
EXECUTE FUNCTION task("alter chunk online","dbspace4","/dev/raw_dev1","0");
```

Related reference:

"onspaces -s: Change status of a mirrored chunk" on page 19-26

alter logmode argument: Change the database logging mode (SQL administration API)

Use the **alter logmode** argument with the **admin()** or **task()** function to change the database logging mode to ANSI, buffered, non-logging, or unbuffered.

Syntax

```
►► EXECUTE FUNCTION admin task ("alter logmode",—"database"—,—"a"  
"b"  
"n"  
"u")  
►►) —; ►►
```

Element	Description	Key Considerations
<i>database</i>	The name of the database with the logging mode that you want to alter.	

Usage

Unlike when you change the database logging mode with the **ondblog** or **ontape** utilities, when you use this function, the database remains accessible, and no level-0 backup is required. Ensure that no other session is active before running this function or it will fail.

Use the **"a"** argument to change the database logging to be ANSI compliant. After you create or convert a database to ANSI mode, you cannot change it back to any of the other logging modes.

Use the **"b"** argument to change the database logging to be buffered, so that transaction information is written to a buffer before it is written to a logical log.

Use the **"n"** argument to change the database logging to be non-logging, so that no database transactions are logged.

Use the **"u"** argument to change the database logging to be unbuffered, so that data is not written to a buffer before it is written to a logical log.

Example

The following example changes the logging mode of a database named **employee** to unbuffered logging:

```
EXECUTE FUNCTION task("alter logmode","employee","u");
```


Related concepts:
Chapter 13, “The onlog Utility,” on page 13-1

alter plog argument: Change the physical log (SQL administration API)

Use the **alter plog** argument with the **admin()** or **task()** function to change the location and size of the physical log.

Syntax

►►EXECUTE FUNCTION admin (—"alter plog"—,—"dbspace",—"size",—)
task,—);

Element	Description	Key Considerations
dbspace	The location of the physical log.	The space allocated for the physical log must be contiguous.
size	The size, specified in kilobytes, of the physical log.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

To change only the size, specify the current dbspace of the physical log.

This function is equivalent to the **onparams -p** command.

Example

The following example moves the physical log to a dbspace called **phsdb**s:

EXECUTE FUNCTION task ("alter plog","phsdb","49 M");

Related reference:

“onparams -p: Change physical-log parameters” on page 16-3

archive fake argument: Perform an unrecorded backup (SQL administration API)

Use the **archive fake** argument with the **admin()** or **task()** function to perform a backup operation to clone the data in a server without creating a persistent backup that could be used to perform a restore.

Syntax

►►EXECUTE FUNCTION admin (—"archive fake"—);
task

Usage

Use this function to populate the secondary server in a High-Availability Data Replication pair.

This function is equivalent to running the **ontape** command with the **-F** option.

Example

The following example starts an unrecorded backup:

```
EXECUTE FUNCTION task("archive fake");
```

cdr argument: Administer Enterprise Replication (SQL administration API)

Use the **cdr** argument with the **admin()** or **task()** function to administer Enterprise Replication.

Syntax

►►—EXECUTE FUNCTION admin
task (—'cdr—*command_name*' (1)
, —'*option*') — ; ►►

Notes:

- 1 Maximum of six option arguments.

Element	Description	Key Considerations
<i>command_name</i>	The name of a cdr command.	You cannot include any hyphens, flags, or other constraining options to <i>command_name</i> that the cdr command-line utility requires. You cannot use abbreviations.
<i>option</i>	One or more elements of the cdr command-line options to the <i>command_name</i> .	The elements must be delimited by quotation marks. Also, include (in the correct order) any hyphens, flags, or other elements of cdr command-line options that the <i>command_name</i> requires. You can use abbreviations.

Usage

Use these functions to produce the same effect as with the **cdr** command-line utility to manage Enterprise Replication.

The SQL administration API supports **cdr** commands used to administer Enterprise Replication. The following commands for monitoring Enterprise Replication are not supported:

- **cdr list grid**
- **cdr list replicate**
- **cdr list replicateset**
- **cdr list server**
- **cdr list template**
- **cdr stats recv**
- **cdr stats rqm**
- **cdr -V**
- **cdr view**

The first argument must include only the **cdr** command names exactly as specified in the appendix for the **cdr** utility in the *IBM Informix Enterprise Replication Guide*, such as **cdr define server**. Command names are case-sensitive and abbreviations (such as **cdr sto replset** instead of **cdr stop replicateset**) are **not** supported. The SQL administration API does not perform any validation before passing the parameters to the **cdr** utility.

The second and any following arguments include the command options. The options can be specified in one or up to six arguments.

The following example illustrates the use of the SQL administration API to define an Enterprise Replication server:

```
EXECUTE FUNCTION task ( 'cdr define server', '--connect=g_amsterdam
--ats=/local0/er/ats --ris=/local0/er/ris --init g_amsterdam' );
```

The following example shows how the options can be spread over several arguments; the above statement can also be written as:

```
EXECUTE FUNCTION task( 'cdr define server',
'--connect=g_amsterdam',
'--ats=/local0/er/ats',
'--ris=/local0/er/ris',
'--init g_amsterdam' );
```

The following example shows double quoted strings within an argument:

```
EXECUTE FUNCTION task('cdr change replicate',
'-d repl_1 -"db1@server1:antonio.table1" "db2@server2:carlo.table2"');
```

check data argument: Check data consistency (SQL administration API)

Use the **check data** argument with the **admin()** or **task()** function to check all pages in the specified partition for consistency, excluding blobpages and sbpages.

Syntax

```
►►EXECUTE FUNCTION admin task ("check data",—"partition_number"—);►►
```

Element	Description	Key Considerations
<i>partition_number</i>	The partition number in which to check the data.	Find the partition numbers in the partnum column of the systables system catalog table.

Usage

Run this function to check entries in the bitmap page against the pages to verify mapping.

This function is equivalent to the **oncheck -cd** command.

Example

The following example checks the consistency of all pages in the partition with a number of 1048611:

```
EXECUTE FUNCTION task("check data","1048611");
```

Related reference:

“**oncheck -cd** and **oncheck -cD** commands: Check pages” on page 8-8

check extents argument: Check extent consistency (SQL administration API)

Use the **check extents** argument with the **admin()** or **task()** function to verify that the extents on disk correspond to the current control information.

Syntax

```
►►—EXECUTE FUNCTION—admin—————►  
                          task  
  
►—(—"check extents"—,—dbspace_number—)—;—————►►
```

Element	Description	Key Considerations
<i>dbspace_number</i>	The number of the dbspace to check.	

Usage

Run this function to check each chunk-free list and corresponding free space and each tblspace extent. If you do not specify a dbspace number, all dbspaces are checked. The function checks dbspaces, blobspaces, smart-large-object extents, and user-data and metadata information in sbpace chunks.

This function is equivalent to the **oncheck -ce** command.

Example

The following example checks the extents in the dbspace with the number 2:

```
EXECUTE FUNCTION task("check extents",2);
```

Related reference:

“**oncheck -ce, -pe**: Check the chunk-free list” on page 8-10

check partition argument: Check partition consistency (SQL administration API)

Use the **check partition** argument with the **admin()** function to print a tblspace report; or use the **check partition** argument with the **task()** function to print a count of rows for the specified partition.

Syntax

```

▶▶EXECUTE FUNCTION [admin | task]
▶(—"check partition"—,—"partition_number"—)—;

```

Element	Description	Key Considerations
<i>partition_number</i>	The partition number in which to check the data.	Find the partition numbers in the partnum column of the systables system catalog table.

Usage

The **check partition** argument with the **admin()** is equivalent to the **oncheck -pt** command.

Example

The following example prints a count of rows for the partition with a number of 1048611:

```
EXECUTE FUNCTION task("check partition","1048611");
```

Related reference:

“oncheck -pt and -pT: Display tblspaces for a Table or Fragment” on page 8-19

checkpoint argument: Force a checkpoint (SQL administration API)

Use the **checkpoint** argument with the **admin()** or **task()** function to force a checkpoint.

Syntax

```

▶▶EXECUTE FUNCTION [admin | task] (—"checkpoint"—,—[
    "hard"
    "block"
    "norm"
    "unblock"
])—;

```

Usage

This function forces a checkpoint that flushes the buffers to disk. You can use this function to force a checkpoint if the most recent checkpoint record in the logical log was preventing the logical-log file from being freed (status U-B-L).

Use the **block** argument to prevent the database server from processing any transactions. Use this option to perform an external backup on Informix. While the database server is blocked, users cannot access it, except in read-only mode. No transactions can complete until the database server is unblocked.

Use the **hard** argument to force a blocking checkpoint. This is the default.

Use the **norm** argument to force a nonblocking checkpoint.

Use the **unlock** argument to unblock the database server. When the database server is unblocked, data transactions and normal database server operations can resume. Use this option after you complete an external backup on Informix.

This function is equivalent to the **onmode -c** command.

Example

The following example starts a blocking checkpoint:

```
EXECUTE FUNCTION task("checkpoint","block");
```

Related reference:

“onmode -c: Force a checkpoint” on page 14-4

clean sbospace argument: Release unreferenced smart large objects (SQL administration API)

Use the **clean sbospace** argument with the **admin()** or **task()** function to release any unreferenced BLOB or CLOB objects from the sbospace.

Syntax

```
►►EXECUTE FUNCTION 

|       |
|-------|
| admin |
| task  |

 ("clean sbospace",—"sboospace"—);◄◄
```

Element	Description	Key Considerations
sboospace	The name of the sbospace to clean.	

Usage

This function is equivalent to the **onspaces -cl** command.

Example

The following example cleans an sbospace named **sbsp1**:

```
EXECUTE FUNCTION task("clean sbospace","sbsp1");
```

Related reference:

“onspaces -cl: Clean up stray smart large objects in sbospaces” on page 19-18

create blobospace argument: Create a blobospace (SQL administration API)

Use the **create blobospace** argument with the **admin()** or **task()** function to create a blobospace.

Syntax

```
►►EXECUTE FUNCTION 

|       |
|-------|
| admin |
| task  |

 (◄◄
```

```

▶ "create" ┌──────────┐ "blobspace" ──,─"blobspace"─,─"path"──────────▶
           └─with_check─┘

▶ ┌───────────────────────────────────────────────────────────────────────────────────┐)─;──────────▶
  │,─"size"──────────────────────────────────────────────────────────────────────────┘
  │┌──────────┐
  │└─"offset"─┘
  │┌──────────┐
  │└─"page_size"─┘

```

Element	Description	Key Considerations
<i>blobspace</i>	The name of the blobspace to be created.	
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the device to reach the initial chunk of the new blobspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>page_size</i>	The blobspace blobpage size.	<p>You specify the size of a blobpage in multiples of a disk page.</p> <p>Valid page sizes depend on the default page size for the computer:</p> <ul style="list-style-type: none"> • 2 KB default page size: 2, 4, 6, 8, 10, 12, or 16 KB • 4 KB default page size: 4, 8, 12, or 16 KB <p>For more information, see blobpage size considerations in the <i>IBM Informix Performance Guide</i>.</p>
<i>path</i>	The disk partition or device of the initial chunk of the blobspace that you are creating.	
<i>size</i>	The size, in kilobytes, of the initial chunk of the new blobspace.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

Use the **create with_check blobspace** argument to check the specified path name and return an error if it does not exist.

This function is equivalent to the **onspaces -c -b** command.

Example

The following example creates a blobspace that has a size of 20 MB with an offset of 0 and page size of 2 KB:

```
EXECUTE FUNCTION task ("create with_check blobspace","blobs3",
"$INFORMIXDIR/WORK/blobs3","20 M","0","2");
```

Related reference:

“onspaces -c -b: Create a blobspace” on page 19-5

“Avoid overwriting a chunk” on page 19-27

create blobspace from storagepool argument: Create a blobspace from the storage pool (SQL administration API)

Use the **create blobspace from storagepool** argument with the **admin()** or **task()** function to create a blobspace from an entry from the storage pool.

Syntax

```
►—EXECUTE FUNCTION admin  
                  task (—"create blobspace from storagepool"—  
          ,—"space_name"—,—"size"—,—"blobpage_size"—,—"mirroring_flag"—) —; ►
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the blobspace.	The blobspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character.
<i>size</i>	The size, in kilobytes, of the initial chunk of the new blobspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>blobpage_size</i>	The blobpage size, specified in terms of <i>page_unit</i> , the number of disk pages per blobpage	The page size is optional. However if you specify 1 for mirroring, you must also specify a page size.
<i>mirroring_flag</i>	Either: 1 = mirroring 0 = no mirroring	The mirroring flag is optional.

Examples

The following command creates a mirrored blobspace named blobspace1. The new blobspace has a size of 100 gigabytes and a blobpage size of 100 pages.

```
EXECUTE FUNCTION task("create blobspace from storagepool", "blobspace1", "100 GB",  
"100", "1");
```

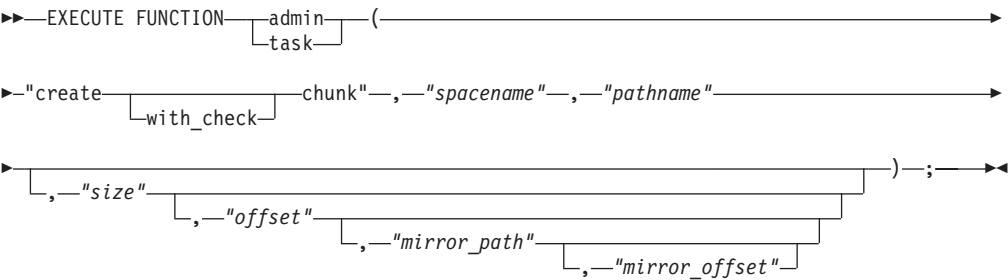
The following command creates an unmirrored blobspace named blobspace2 with the default blobpage size, so a blobpage size is not specified:

```
EXECUTE FUNCTION task("create blobspace from storagepool", "blobspace2", "5000");
```


create chunk argument: Create a chunk (SQL administration API)

Use the **create chunk** argument with the **admin()** or **task()** function to create a chunk in a dbspace or in a blobspace.

Syntax



Element	Description	Key Considerations
<i>mirror_path</i>	The path to the mirror chunk. If you are adding a chunk to a mirrored storage space, you must also add a mirror chunk.	
<i>mirror_offset</i>	The location of the mirror chunk.	
<i>offset</i>	The location of the new chunk.	
<i>pathname</i>	The path of the added disk space.	
<i>size</i>	The amount of disk space to add in kilobytes.	See "admin() and task() Argument Size Specifications" on page 21-2.
<i>spacename</i>	The name of the dbspace, blobspace, or sbpace to which you are adding disk space.	

Usage

Use the **create with_check chunk** argument to check the specified path name and return an error if it does not exist.

This function is equivalent to the **onspaces -a** command.

Example

The following example adds a 5 MB chunk of raw disk space, at an offset of 5200 kilobytes, to a dbspace named **dbspc3**:

```
EXECUTE FUNCTION task("create chunk", "dbspc3", "\\.\e:", "5120", "5200");
```

The following example adds a 10 MB mirror chunk to a blobspace named **blobsp3** with an offset of 200 kilobytes for both the primary and mirror chunks:

```
EXECUTE FUNCTION task("create with_check chunk","blobsp3","/dev/raw_dev1","10240",
"200","/dev/raw_dev2","200");
```

Related reference:

“onspaces -a: Add a chunk to a dbspace or blobspace” on page 19-2

“Avoid overwriting a chunk” on page 19-27

“onspaces -a: Add a chunk to an sbospace” on page 19-3

create chunk from storagepool argument: Create a chunk from the storage pool (SQL administration API)

Use the **create chunk from storagepool** argument with the **admin()** or **task()** function to manually create a chunk from an entry in the storage pool.

Syntax

```
➤➤EXECUTE FUNCTION admin task ("—create chunk from storagepool—"➤➤
➤➤,—"space_name"—,—"size"—)—;➤➤
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the storage space to which you are adding the chunk.	
<i>size</i>	The size, in kilobytes, of the initial chunk.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

You can also use an SQL administration API command with the **modify space expand** argument to manually create a chunk from the storage pool and add the chunk to the specified storage space. However, if the space has extendable chunks, Informix might extend a chunk instead of creating a new one. Unlike the **modify space expand** argument, the **create chunk from storagepool** argument forces Informix to add a chunk.

Example

The following command adds a chunk to the dbspace named logdbs. The new chunk has a size of 200 megabytes.

```
EXECUTE FUNCTION task("create chunk from storagepool", "logdbs", "200 MB");
```

Related reference:

“modify space expand argument: Expand the size of a space (SQL administration API)” on page 21-68

create database argument: Create a database (SQL administration API)

Use the **create database** argument with the **admin()** or **task()** function to create a database.

Syntax

```
►► EXECUTE FUNCTION {admin|task} ("create database"
    with {log|buffered log|log mode ansi}
    {nlscase insensitive|nlscase sensitive}
    ,—"database_name"
    [,—"dbspace"
    [,—"locale"
    )—;
```

Element	Description	Key Considerations
<i>database_name</i>	The name of the database.	
<i>dbspace</i>	The name of the dbspace to store the data for this database. The default is the root dbspace.	The dbspace must already exist on the database server.
<i>locale</i>	The locale associated with the database.	

Usage

This function is equivalent to the CREATE DATABASE statement.

Examples

The following example creates the database named demodbs with unbuffered logging:

```
EXECUTE FUNCTION task("create database with log","demodbs");
```

The following example creates a database that is not case-sensitive named demodbs2 with ANSI compliant logging in the dbspace named dataspace1:

```
EXECUTE FUNCTION task("create database with log mode ansi nlscase insensitive",
    "demodbs2","dataspace1");
```

The following example creates a database named demodbs3 with a French-Canadian locale in the dbspace name dataspace1:

```
EXECUTE FUNCTION TASK ("create database","demodbs3","dataspace1","fr_ca.8859-1");
```










Related reference:

 CREATE DATABASE statement (SQL Syntax)

create dbspace argument: Create a dbspace (SQL administration API)

Use the **create dbspace** argument with the **admin()** or **task()** function to create a dbspace.

Syntax

```
EXECUTE FUNCTION   ("create  dbspace"        ) 
```

Element	Description	Key Considerations
<i>dbspace</i>	The name of the dbspace to be created.	
<i>first_extsize</i>	The size, in kilobytes, of the first extent for the tblspace .	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>next_extsize</i>	The size, in kilobytes, of the next extents in the tblspace .	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the device to reach the initial chunk of the new dbspace.	
<i>pagesize</i>	The non-default page size, in kilobytes, for the new dbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>pathname</i>	The disk partition or device of the initial chunk of the dbspace that you are creating.	Valid page sizes depend on the default page size for the computer: <ul style="list-style-type: none">• 2 KB default page size: 2, 4, 6, 8, 10, 12, or 16 KB• 4 KB default page size: 4, 8, 12, or 16 KB
<i>size</i>	The size, in kilobytes, of the initial chunk of the new dbspace. The size is rounded to a multiple of the page size.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

Use the **create with_check dbspace** argument to check the specified path name and return an error if it does not exist.

This function is equivalent to the **onspaces -c -d** command.

Example

The following example creates a dbspace that has a size of 20 MB with an offset of 0.

```
EXECUTE FUNCTION task ("create dbspace", "dbspace3",  
"$INFORMIXDIR/WORK/dbspace3", "20 M", "0");
```

Related reference:


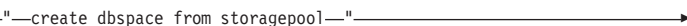

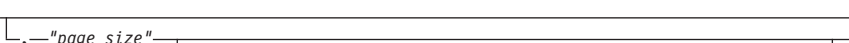
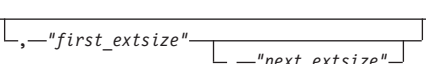
“onspaces -c -d: Create a dbspace” on page 19-7

“Avoid overwriting a chunk” on page 19-27

create dbspace from storagepool argument: Create a dbspace from the storage pool (SQL administration API)

Use the **create dbspace from storagepool** argument with the **admin()** or **task()** function to create a dbspace from an entry in the storage pool.

Syntax

```
► EXECUTE FUNCTION  "create dbspace from storagepool"    
►,—"space_name"—,"size"—,    
►  )—; 
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the dbspace.	The dbspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character.
<i>size</i>	The size of the initial chunk of the new dbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>page_size</i>	The non-default page size, in kilobytes, for the new dbspace.	The page size is optional. However if you specify 1 for mirroring, you must also specify a page size.
<i>mirroring_flag</i>	Either: 1 = mirroring 0 = no mirroring	The mirroring flag is optional.
<i>first_extent</i>	The size, in kilobytes, of the first extent for the tblspace .	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>next_extent</i>	The size, in kilobytes, of the next extent for the tblspace .	See “admin() and task() Argument Size Specifications” on page 21-2.

Examples

The following command creates a mirrored dbspace named dbspace3. The new dbspace has a size of 1 gigabyte, a page size of 6 kilobytes, a tblspace **tblspace** first extent size of 200 kilobytes, and a next extent size of 400 kilobytes.

```
EXECUTE FUNCTION task("create dbspace from storagepool",  
  "dbspace3", "1 GB", "6", "1", "200", "400");
```

The following command creates an unmirrored dbspace named dbspace8. The size of the new dbspace is 5 megabytes. The page size is not specified, because the new dbspace has the default page size .

```
EXECUTE FUNCTION task("create dbspace from storagepool",  
  "dbspace8", "50000");
```

create sbpace argument: Create an sbpace (SQL administration API)

Use the **create sbpace** argument with the **admin()** or **task()** function to create an sbpace.

Syntax

```
►►EXECUTE FUNCTION admin  
task (—————►  
  
►"create with_check sbpace"—,—"sbspace"—,—"pathname"—————►  
  
►,—"size"  
,—"offset")—;—————►◄
```

Element	Description	Key Considerations
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the device to reach the initial chunk of the new sbpace.	
<i>pathname</i>	The disk partition or unbuffered device of the initial chunk of the sbpace.	
<i>sbspace</i>	The name of the sbpace to be created.	
<i>size</i>	The size, in kilobytes, of the initial chunk of the new sbpace.	See "admin() and task() Argument Size Specifications" on page 21-2.

Usage

Use the **create with_check sbpace** argument to check the specified path name and return an error if it does not exist.

This function is equivalent to the **onspaces -c -S** command.

Example

The following example creates a new sbspace that has a size of 20 MB with an offset of 0:

```
EXECUTE FUNCTION task ("create sbspace", "sbspace2",  
"$INFORMIXDIR/WORK/sbspace2", "20 M", "0");
```

Related reference:

“onspaces -c -S: Create an sbspace” on page 19-11



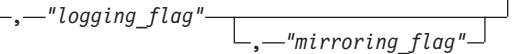
“Avoid overwriting a chunk” on page 19-27

“create tempsbspace argument: Create a temporary sbspace (SQL administration API)” on page 21-39

create sbspace from storagepool argument: Create an sbspace from the storage pool (SQL administration API)

Use the **create sbspace from storagepool** argument with the **admin()** or **task()** function to create an sbspace from an entry from the storage pool.

Syntax

```
►► EXECUTE FUNCTION  ("create sbspace from storagepool"  )  ; ►►
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the sbspace.	The sbspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character.
<i>size</i>	The size, in kilobytes, of the initial chunk of the new sbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>logging_flag</i>	Either: 1 = logging 0 = no logging	The logging flag is optional. However if you specify 1 for mirroring, you must also specify a logging flag.
<i>mirroring_flag</i>	Either: 1 = mirroring 0 = no mirroring	The mirroring flag is optional.

Examples

The following command creates a mirrored and logged sbspace named sbspace1. The new sbspace has a size of 240 megabytes.

```
EXECUTE FUNCTION task("create sbspace from storagepool", "sbspace1",  
"240 MB", "1", "1");
```

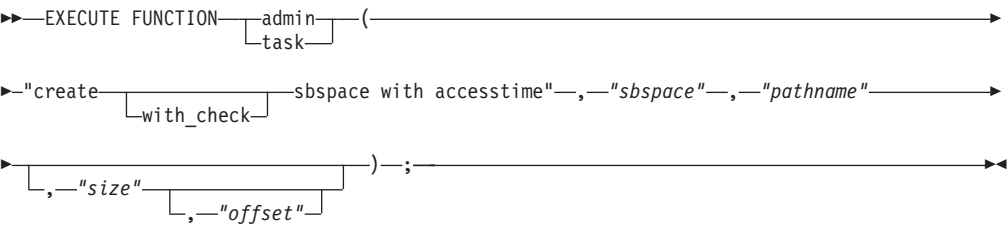
The following command creates an unmirrored and unlogged sbospace named sbospace2. This sbospace has a size of 5 gigabytes.

```
EXECUTE FUNCTION task("create sbospace from storagepool", "sbospace2", "5 GB");
```

create sbospace with accesstime argument: Create an sbospace that tracks access time (SQL administration API)

Use the **create sbospace with accesstime** argument with the **admin()** or **task()** function to create an sbospace that tracks the time of access for all smart large objects stored in the sbospace.

Syntax



Element	Description	Key Considerations
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the device to reach the initial chunk of the new sbospace.	
<i>pathname</i>	The disk partition or unbuffered device of the initial chunk of the sbospace.	
<i>sbospace</i>	The name of the sbospace to be created.	
<i>size</i>	The size, in kilobytes, of the initial chunk of the new sbospace.	See "admin() and task() Argument Size Specifications" on page 21-2.

Usage

Use the **create with_check sbospace** argument to check the specified path name and return an error if it does not exist.

This function is equivalent to the **onspaces -c -S** command for creating an sbospace and using the **set sbospace accesstime** argument with the **admin()** or **task()** function to start tracking the time of access for all smart large objects stored in the sbospace.

Example

The following example creates a new sbospace that tracks access time. This sbospace has a size of 20 MB with an offset of 0:

```
EXECUTE FUNCTION task ("create sbospace with accesstime","sbospace4",  
"$INFORMIXDIR/WORK/sbospace4","20 M","0");
```


Related reference:

“set sbspace accesstime argument: Control access time tracking (SQL administration API)” on page 21-109

create sbspace with log argument: Create an sbspace with transaction logging (SQL administration API)

Use the **create sbspace with log** argument with the **admin()** or **task()** function to create an sbspace with transaction logging turned on.

Syntax

```
►►—EXECUTE FUNCTION admin  
                          task (—————►  
  
►—"create with_check sbspace with log"—,—"sbspace"—,—"pathname"—►  
                                          size offset )—;—————►
```

Element	Description	Key Considerations
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the device to reach the initial chunk of the new sbspace.	
<i>pathname</i>	The disk partition or unbuffered device of the initial chunk of the sbspace.	
<i>sbspace</i>	The name of the sbspace to be created.	
<i>size</i>	The size, in kilobytes, of the initial chunk of the new sbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

Use the **create with_check sbspace** argument to check the specified path name and return an error if it does not exist.

This function is equivalent to the **onspaces -c -S** command to create an sbspace with the option for turning logging on.


Example

The following example creates a new sbspace with transaction logging turned on. This sbspace has a size of 20 MB with an offset of 0:

```
EXECUTE FUNCTION task ("create sbspace with log","sbspace2",  
"$INFORMIXDIR/WORK/sbspace2","20 M","0");
```

Related reference:










“onspaces -c -S: Create an sbspace” on page 19-11

 Sbspace logging (Administrator's Guide)

create tempdbspace argument: Create a temporary dbspace (SQL administration API)

Use the **create tempdbspace** argument with the **admin()** or **task()** function to create a temporary dbspace.

Syntax

```
→EXECUTE FUNCTION  ("create  tempdbspace"        ) ; →
```

Element	Description	Key Considerations
<i>dbspace</i>	The name of the temporary dbspace to be created.	
<i>first_extsize</i>	The size, in kilobytes, of the first extent for the tblspace tblspace .	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>next_extsize</i>	The size, in kilobytes, of the next extents in the tblspace tblspace .	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the device to reach the initial chunk of the new temporary dbspace.	
<i>page_size</i>	The non-default page size, in kilobytes, for the new temporary dbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>pathname</i>	The disk partition or device of the initial chunk of the temporary dbspace that you are creating.	Valid page sizes depend on the default page size for the computer: <ul style="list-style-type: none">• 2 KB default page size: 2, 4, 6, 8, 10, 12, or 16 KB• 4 KB default page size: 4, 8, 12, or 16 KB
<i>size</i>	The size, in kilobytes, of the initial chunk of the new temporary dbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

Use the **create with_check tempdbspace** argument to check the specified path name and return an error if the path does not exist.

This function is equivalent to the **onspaces -c -d -t** command.

Example

The following example creates a temporary dbspace that has a size of 20 MB with an offset of 0:

```
EXECUTE FUNCTION task("create tempdbspace","tempdbspace3",
"$INFORMIXDIR/WORK/tempdbspace3","20 M","0");
```

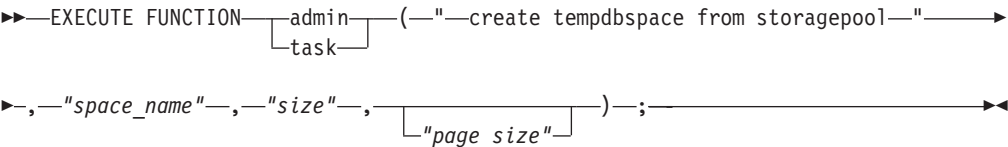
Related reference:

“onspaces -c -d: Create a dbspace” on page 19-7

create tempdbspace from storagepool argument: Create a temporary dbspace from the storage pool (SQL administration API)

Use the **create tempdbspace from storagepool** argument with the **admin()** or **task()** function to create a temporary dbspace from an entry from the storage pool.

Syntax



Element	Description	Key Considerations
<i>space_name</i>	The name of the temporary dbspace.	
<i>size</i>	The size, in kilobytes, of the initial chunk of the new temporary dbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.
<i>page_size</i>	The non-default page size, in kilobytes, for the new temporary dbspace.	The page size is optional.

Example

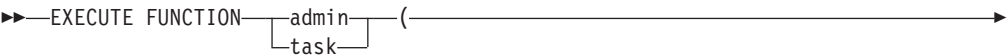
The following command creates a temporary dbspace named tempdbspace1. The new dbspace has a size of 1 gigabyte and a page size of 12 kilobytes.

```
EXECUTE FUNCTION task("create tempdbspace from storagepool", "tempdbspace1",
"1 GB", "12");
```

create tempsbspace argument: Create a temporary sbospace (SQL administration API)

Use the **create sbospace** argument with the **admin()** or **task()** function to create an sbospace.

Syntax



```

▶ "create" ┌──────────┐ tempsbbspace" ─, ─ "sbspace" ─, ─ "pathname" ───────────▶
           │with_check│
           └──────────┘

▶ ┌──────────┐ ─, ─ "size" ─, ─ "offset" ─ ───────────▶
  │──────────┘
  └──────────┘
  ) ─; ───────────▶

```

Element	Description	Key Considerations
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the device to reach the initial chunk of the new temporary sbspace.	
<i>pathname</i>	The disk partition or unbuffered device of the initial chunk of the temporary sbspace.	
<i>sbspace</i>	The name of the temporary sbspace to be created.	
<i>size</i>	The size, in kilobytes, of the initial chunk of the new temporary sbspace.	See "admin() and task() Argument Size Specifications" on page 21-2.

Usage

Use the **create with_check sbspace** argument to check the specified path name and return an error if it does not exist.

This function is equivalent to the **onspaces -c -S** command with the **-t** option for creating a temporary sbspace.

Example


The following example creates a temporary sbspace that has a size of 20 MB with an offset of 0:

```

EXECUTE FUNCTION task ("create tempsbbspace","tempsbbspace3",
"$INFORMIXDIR/WORK/tempsbbspace3","20 M","0");

```

Related concepts:

 Temporary sbspaces (Administrator's Guide)

Related reference:

“create sbspace argument: Create an sbspace (SQL administration API)” on page 21-34

“onspaces -c -S: Create an sbspace” on page 19-11

create tempsbspace from storagepool argument: Create a temporary sbspace from the storage pool (SQL administration API)

Use the **create tempsbspace from storagepool** argument with the **admin()** or **task()** function to create a temporary sbspace from an entry from the storage pool.

Syntax

```
►►EXECUTE FUNCTION admin task (—"create tempsbspace from storagepool"—►  
►,—"space_name"—,—"size"—)►;►
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the temporary sbspace.	The temporary sbspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the \$ character.
<i>size</i>	The size, in kilobytes, of the initial chunk of the new sbspace.	See “admin() and task() Argument Size Specifications” on page 21-2.

Example

The following command creates a temporary sbspace named tempsbspace5. The temporary sbspace has a size of 240 megabytes.

```
EXECUTE FUNCTION task("create tempsbspace from storagepool",  
"tempsbspace5", "240 MB");
```

defragment argument: Dynamically defragment partition extents



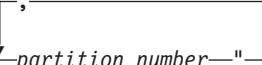
Use the **defragment** argument with the **admin()** or **task()** function to defragment tables or indexes to merge non-contiguous extents.

Defragmenting a table brings data rows closer together to avoid partition header page overflow problems, and can improve performance.

Before you defragment a partition you should review the limitations and other important considerations.

Syntax

You can specify either the defragment argument or defragment partnum argument using the following syntax:

```
►►EXECUTE FUNCTION 
►(-"defragment"—,—"database—:—owner—.—table—"—);
►►EXECUTE FUNCTION 
►(-"defragment partnum"—, );
```

Element	Description	Key considerations
<i>database</i>	Name of the database that includes the table or index that you want to defragment.	
<i>owner</i>	User ID of the owner of the table.	
<i>table</i>	Name of the table to defragment.	
<i>partition_number</i>	One or more partition numbers to defragment.	Use a comma-separated list of partition numbers to specify more than one partition.

Usage

Use the defragment argument to defragment specific tables. Use the defragment partnum argument to defragment one or more specific disk partitions.

Information about defragmentation is stored in shared memory. Use the **oncheck -pt and pT** command to display information about the number of extents for a specific table or fragment. Use the **onstat -g defragment** command to display information about active defragment requests. See the *IBM Informix Administrator's Reference* for more information about the **oncheck -pt and pT** and **onstat -g defragment** commands.

If the defragment request reduces the number of extents by at least 1 extent, the request returns 0 (success), even if there are many extents in the partition.

If a partition has a single extent, the defragment request returns 0 to indicate that the request was a success, even though no extents were merged.

Examples

To defragment the customer table in the stores_demo database, use either of the following functions:

```
EXECUTE FUNCTION task("defragment","stores_demo:informix.customer");
EXECUTE FUNCTION admin("defragment","stores_demo:informix.customer");
```

To defragment an index, you must specify the partition number for the index, as in these two function examples:

```
EXECUTE FUNCTION task("defragment partnum","2097154");
EXECUTE FUNCTION admin("defragment partnum","2097154");
```

To defragment a list of partitions, use either of the following functions:

```
EXECUTE FUNCTION task("defragment partnum", "16777217,28477346");
EXECUTE FUNCTION admin("defragment partnum", "16777217,28477346");
```

drop blobspace argument: Drop a blobspace (SQL administration API)

Use the **drop blobspace** argument with the **admin()** or **task()** function to drop the specified blobspace.

Syntax

```
►►EXECUTE FUNCTION admin task (—"drop blobspace"—,—"blobspace_name"—►►
►-)—;—►◄
```

Element	Description	Key Considerations
<i>blobspace_name</i>	The name of the blobspace to drop.	Must be an existing blobspace. Before you drop a blobspace, drop all tables that include a TEXT or BYTE column that references the blobspace.

Usage

This function is equivalent to the **onspaces -d** command.

Example

The following example drops the blobspace named **blobspace3**:

```
EXECUTE FUNCTION task("drop blobspace","blobspace3");
```




Related reference:

"onspaces -d: Drop a blobspace, dbspace, extspace, or sbspace" on page 19-20

drop blobspace to storagepool argument: Return space from an empty blobspace to the storage pool (SQL administration API)

Use the **drop blobspace to storagepool** argument with the **admin()** or **task()** function to return the space from an empty blobspace to the storage pool.

Syntax

```
►►—EXECUTE FUNCTION  ("drop blobspace to storagepool"    
►,—"space_name"—);— 
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the empty blobspace.	

Example



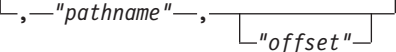

The following command drops an empty blobspace named blob2 and adds all of the freed space to the storage pool.

```
EXECUTE FUNCTION task("drop blobspace to storagepool", "blob2");
```

drop chunk argument: Drop a chunk (SQL administration API)

Use the **drop chunk** argument with the **admin()** or **task()** function to drop the specified chunk from a dbspace, blobspace, or sbspace.

Syntax

```
►►—EXECUTE FUNCTION  ("drop chunk"    
►,—"space_name"  )—;— 
```

Element	Description	Key Considerations
<i>offset</i>	The offset, in kilobytes, into the disk partition or into the unbuffered device to reach the initial chunk of the dbspace, blobspace, or sbspace that you are dropping.	The starting offset, an unsigned integer, must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 TB. Also see "admin() and task() Argument Size Specifications" on page 21-2.

Element	Description	Key Considerations
<i>pathname</i>	The disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you are dropping.	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server.
<i>space_name</i>	The name of the dbspace, sbspace, or blobspace from which to drop a chunk.	You can drop a chunk from a dbspace, temporary dbspace, or sbspace when the database server is online or quiescent. You can drop a chunk from a blobspace only when the database server is in quiescent mode.

Usage

This function is equivalent to the **onspaces -d** command.

Example

The following example drops a chunk at an offset of 5200 kilobytes from a dbspace named **dbspc3**:

```
EXECUTE FUNCTION task("drop chunk", "dbspc3", "\\.\e:", "5200");
```

Related reference:

“onspaces -d: Drop a chunk in a dbspace, blobspace, or sbspace” on page 19-18

drop chunk to storagepool argument: Return space from an empty chunk to the storage pool (SQL administration API)

Use the **drop chunk to storagepool** argument with the **admin()** or **task()** function to return the space from an empty chunk to the storage pool.

Syntax

```

▶▶—EXECUTE FUNCTION admin task (—"drop chunk to storagepool"——————▶
▶,—"space_name"—,—"path"—,—"offset"—)—;—————▶▶

```

Element	Description	Key Considerations
<i>space_name</i>	The name of the storage space in which the chunk resides.	
<i>path</i>	The path of the chunk.	

Element	Description	Key Considerations
<i>offset</i>	The offset, in kilobytes, of the chunk.	

Example

The following command drops an empty chunk in a dbspace named bigdbs and adds all of the freed space to the storage pool.

```
EXECUTE FUNCTION task("drop chunk to storagepool", "bigdbs", "/dev/rawdisk23",
"100 KB");
```

drop database argument: Drop a database (SQL administration API)

Use the **drop database** argument with the **admin()** or **task()** function to drop a database.

Syntax

```
►►EXECUTE FUNCTION admin task ("drop database"—,"database_name"—);►►
```

Element	Description	Key Considerations
<i>database_name</i>	The name of the database.	

Usage

This function is equivalent to the DROP DATABASE statement. This function deletes the entire database, including all of the system catalog tables, objects, and data.

Example

The following example drops the database named demodbs:

```
EXECUTE FUNCTION task("drop database","demodbs");
```

Related reference:

[📄 DROP DATABASE statement \(SQL Syntax\)](#)

drop dbspace argument: Drop a dbspace (SQL administration API)

Use the **drop dbspace** argument with the **admin()** or **task()** function to drop the specified dbspace.

Syntax

```
►►EXECUTE FUNCTION admin task ("drop dbspace"—,"dbspace_name"—);►►
```

Element	Description	Key Considerations
<i>dbspace_name</i>	The name of the dbspace to drop.	The dbspace must exist. Before you drop a dbspace, drop all databases and tables that you previously created in the dbspace.

Usage

This function is equivalent to the **onspaces -d** command.

Example

The following example drops the dbspace named **dbspace4**:

```
EXECUTE FUNCTION task("drop dbspace","dbspace4");
```

Related reference:

“onspaces -d: Drop a blobspace, dbspace, extspace, or sbpace” on page 19-20

drop dbspace to storagepool argument: Return space from an empty dbspace to the storage pool (SQL administration API)

Use the **drop dbspace to storagepool** argument with the **admin()** or **task()** function to return the space from an empty dbspace to the storage pool.

Syntax

```

▶▶EXECUTE FUNCTION admin task ("drop dbspace to storagepool" 
▶▶,—"space_name"—) —; 

```

Element	Description	Key Considerations
<i>space_name</i>	The name of the empty dbspace.	

Example

The following command drops an empty dbspace named **db5** and adds all of the freed space to the storage pool.

```
EXECUTE FUNCTION task("drop dbspace to storagepool", "db5");
```

drop log argument: Drop a logical log (SQL administration API)

Use the **drop log** argument with the **admin()** or **task()** function to drop the specified logical log.

Syntax

```

▶▶EXECUTE FUNCTION admin task ("drop log"—,"log_number"—) —; 

```

Element	Description	Key Considerations
<i>log_number</i>	The logical log file number.	The number must be an unsigned integer greater than or equal to 0.

Usage

Use this function to drop a single logical log file.

The database server requires a minimum of three logical-log files at all times. You cannot drop a log file if the database server has only three logical-log files.

Important: Before you can drop any of the first three logical-log files, you must add new logical-log files and run a backup of the logical-log files. The backup must be run using either the **ontape -a** command or the **ontape -c** command. After you add the new logical-log files and run a backup, you can then use **onparams -d -llognum** to delete the first three logical-log files.

The status of the log file determines if the log file can be dropped, and the actions taken by the database server when the log file is dropped:

- If you drop a log file that has never been written to, status is newly Added (**A**), the database server deletes the log file and frees the space immediately.
- If you drop a used log file that has a status of User (**U**) or Free (**F**), the database server marks the log file as Deleted (**D**). After you take a level-0 backup of the dbspaces that contain the log files and the root dbspace, the database server deletes the log file and frees the space.
- You cannot drop a log file that is currently in use (**C**) or contains the last checkpoint record (**L**).

You can obtain the log number from the number field of the **onstat -l** command. The sequence of log numbers might be out of order.

This function is equivalent to the **onparams -d -l lognum** command.

Example

The following example drops the logical log with a file number of 2:

```
EXECUTE FUNCTION task("drop log","2");
```

The following example drops the log for a specific chunk by looking up the log number based on the chunk number:

```
SELECT task("drop log", number) FROM sysmaster:syslogfil WHERE chunk = 1;
```

Related reference:

“onparams -d -l *lognum*: Drop a logical-log file” on page 16-2

drop sbospace argument: Drop an sbospace (SQL administration API)

Use the **drop sbospace** argument with the **admin()** or **task()** function to drop the specified sbospace.

Syntax

```
►►EXECUTE FUNCTION admin task (—"drop sbospace"—,—"sospace_name"—)—;—►►
```

Element	Description	Key Considerations
<i>sospace_name</i>	The name of the sbospace to drop.	The sbospace must exist. Before you drop an sbospace, drop all tables that include a BLOB or CLOB column that references the sbospace.

Usage

This function is equivalent to the **onspaces -d** command.

Example

The following example drops the sbospace named **sospace3**:

```
EXECUTE FUNCTION task("drop dbospace","sospace3");
```

Related reference:

“onspaces -d: Drop a blobospace, dbospace, extspace, or sbospace” on page 19-20

drop sbospace to storagepool argument: Return space from an empty sbospace to the storage pool (SQL administration API)

Use the **drop sbospace to storagepool** argument with the **admin()** or **task()** function to return the space from an empty sbospace to the storage pool.

Syntax

```
►►EXECUTE FUNCTION admin task (—,——"drop sbospace to storagepool"——►►  
  
►—"space_name"—)—;—►►
```

Element	Description	Key Considerations
<i>space_name</i>	The name of the empty sbospace.	

Example

The following command drops an empty sbspace named sbspace8 and adds all of the freed space to the storage pool.

```
EXECUTE FUNCTION task("drop sbspace to storagepool", "sbspace8");
```

drop tempdbspace argument: Drop a temporary dbspace (SQL administration API)

Use the **drop tempdbspace** argument with the **admin()** or **task()** function to drop the specified temporary dbspace.

Syntax

```
►►—EXECUTE FUNCTION admin task (—"drop tempdbspace"—,—"tempdbspace_name"—►►  
►) —; —►►
```

Element	Description	Key Considerations
<i>tempdbspace_name</i>	The name of the temporary dbspace to drop.	The temporary dbspace must exist. Before you drop a temporary dbspace, drop all databases and tables that you previously created in the temporary dbspace.

Usage

This function is equivalent to the **onspaces -d** command.

Example

The following example drops the temporary dbspace named **tdbspace2**:

```
EXECUTE FUNCTION task("drop tempdbspace", "tdbspace2");
```

Related reference:

“onspaces -d: Drop a blobspace, dbspace, extspace, or sbspace” on page 19-20

drop tempdbspace to storagepool argument: Return space from an empty temporary dbspace to the storage pool (SQL administration API)

Use the **drop tempdbspace to storagepool** argument with the **admin()** or **task()** function to return the space from an empty temporary dbspace to the storage pool.

Syntax

```
►►—EXECUTE FUNCTION admin task (—"drop tempdbspace to storagepool"—►►
```

►,—"*space_name*"—)——►

Element	Description	Key Considerations
<i>space_name</i>	The name of the empty temporary dbspace.	

Example

The following command drops an empty temporary dbspace named tempdbs1 and adds all of the freed space to the storage pool.

```
EXECUTE FUNCTION task("drop tempdbspace to storagepool", "tempdbs1");
```

drop tempsbbspace to storagepool argument: Return space from an empty temporary sbbspace to the storage pool (SQL administration API)

Use the **drop tempsbbspace to storagepool** argument with the **admin()** or **task()** function to return the space from an empty temporary sbbspace to the storage pool.

Syntax

►—EXECUTE FUNCTION—

admin
task

—(—"drop tempsbbspace to storagepool"—)——►

►,—"*space_name*"—)——►

Element	Description	Key Considerations
<i>space_name</i>	The name of the empty temporary sbbspace.	

Example

The following command drops an empty temporary sbbspace named tempsbbspace3 and adds all of the freed space to the storage pool.

```
EXECUTE FUNCTION task("drop tempsbbspace to storagepool", "tempsbbspace3");
```

file status argument: Display the status of a message log file (SQL administration API)

Use the **file status** argument with the **admin()** or **task()** function to specify the status of an online, ON-Bar activity, or ON-Bar debug message log file.

Syntax

►—EXECUTE FUNCTION—

admin
task

——►

►—("file status"—,—"*full_path_for_message_log_file*"——)——►

Element	Purpose	Key considerations
<i>full_path_for_message_log_file</i>	Full path name for the online, ON-Bar activity, or ON-Bar debug message log file.	

Example

The following example shows the argument that you can use to display the status of the `/usr/informix/online.log` file:

```
execute function task("file status", "/usr/informix/online.log");
```

The server then displays information such as:

```
(expression)  File name      = /tmp/x
               Is File       = 1
               Is Directory   = 0
               Is Raw Device  = 0
               Is Block Device = 0
               Is Pipe        = 0
               File Size      = 554
               Last Access Time = 11/29/2010 21:55:02
               Last Modified Time = 11/29/2010 21:51:45
               Status Change Time = 11/29/2010 21:51:45
               User Id        = 200
               Group id       = 102
               File Flags     = 33206
```

Related reference:

“message log rotate argument: Rotate the message log file (SQL administration API)” on page 21-63

“message log truncate argument: Delete the contents of a message log file (SQL administration API)” on page 21-64

“message log delete argument: Delete a message log file (SQL administration API)” on page 21-63

ha make primary argument: Change the mode of a secondary server (SQL administration API)

Use the **ha make primary** argument with the **admin()** or **task()** function to change the specified secondary server to a primary or standard server.

Syntax

```
►►EXECUTE FUNCTION admin task ( "ha make primary" "ha make primary force" )
►,—"dbserver"—) —;
```

Element	Description	Key Considerations
<i>dbserver</i>	The name of the database server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

This function has different results depending on the type of secondary server:

- HDR Secondary: The current primary server is shut down and the HDR secondary server is made the primary server.
- RS secondary: The RS secondary server is changed to a standard server.
- SD secondary: The SD secondary server is made the new primary server.

Use the **ha make primary** argument to change an inactive secondary server to a primary server when there is an active connection between them.

Use the **ha make primary force** argument to change an inactive secondary server to a primary server, whether or not a secondary server is connected to it. If the connection is active, the function succeeds, however, if you run the function with the **force** argument on an SD secondary server, the shared disk subsystem can become corrupted.

This function is equivalent to the **onmode -d make primary** command.

Example

The following example converts an HDR secondary server named **ids_stores2** into a primary server:

```
EXECUTE FUNCTION task("ha make primary","ids_stores2");
```

Related reference:

“onmode -d: Set High Availability server characteristics” on page 14-8

ha rss argument: Create an RS secondary server (SQL administration API)

Use the **ha rss** argument with the **admin()** or **task()** function to create a remote standalone (RS) secondary server.

Syntax

```
►► EXECUTE FUNCTION admin ( task ) ►  
  
► "ha rss" , —"primary_server" , —"password" ►◄  
;  
```

Element	Description	Key Considerations
<i>password</i>	A password to set or to change.	The password is used only during the first connection attempt. After the primary and secondary server have connected, the password cannot be changed.

Element	Description	Key Considerations
<i>primary_server</i>	The name of the primary database server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function on a standard server or an HDR secondary server to convert it an RS secondary server.

This function is equivalent to the **onmode -d RSS** command.

Example

The following example converts a standard server into an RS secondary server with a primary server named **ids_stores**:

```
EXECUTE FUNCTION task("ha rss","ids_stores");
```

Related reference:

“onmode -d: Set High Availability server characteristics” on page 14-8

ha rss add argument: Add an RS secondary server to a primary server (SQL administration API)

Use the **ha rss add** argument with the **admin()** or **task()** function to associate a primary server with a remote standalone (RS) secondary server.

Syntax

```

▶▶ EXECUTE FUNCTION admin ( task ) (
"ha rss add" , "secondary_server" " , "password" " ) ;

```

Element	Description	Key Considerations
<i>password</i>	The password to set or to change.	The password is used only during the first connection attempt. After the primary and secondary server have connected, the password cannot be changed.
<i>secondary_server</i>	The name of the database server to convert to an RS secondary server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function from an established primary server to create an RS secondary server and register the RS secondary server name in the **syssha** database.

This function is equivalent to the **onmode -d add RSS** command.

Example

The following example associates a server named **ids_stores2** as an RS secondary server with the primary server:

```
EXECUTE FUNCTION task("ha rss add","ids_stores2");
```

Related reference:

“onmode -d: Set High Availability server characteristics” on page 14-8

ha rss change argument: Change the password of an RS secondary server (SQL administration API)

Use the **ha rss change** argument with the **admin()** or **task()** function to change the connection password for the specified RS secondary server.

Syntax

```
►►—EXECUTE FUNCTION admin task (—————►  
►—"ha rss change"—,"secondary_server"— , —"password"—)——;—————►◄
```

Element	Description	Key Considerations
<i>password</i>	The password to set or to change.	The password is used only during the first connection attempt. After the primary and secondary server have connected, the password cannot be changed.
<i>secondary_server</i>	The name of the database server to convert to an RS secondary server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function on an established primary server to change the password for the connection between the primary and secondary server.

This function is equivalent to the **onmode -d change RSS** command.

Example

The following example changes the password for the RS secondary server to secure:

EXECUTE FUNCTION task("ha rss change","ids_stores2","secure");

Related reference:

“onmode -d: Set High Availability server characteristics” on page 14-8

ha rss delete argument: Delete an RS secondary server (SQL administration API)

Use the **ha rss delete** argument with the **admin()** or **task()** function to stop replication and delete the RS secondary server.

Syntax

►►EXECUTE FUNCTION admin (—"ha rss delete"—,"secondary_server"►)
task
►) ; ►►

Element	Description	Key Considerations
<i>secondary_server</i>	The name of the database server to convert to an RS secondary server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function from an established primary server to stop replication, delete the RS secondary server, and convert the RS secondary server to a standard server.

This function is equivalent to the **onmode -d delete RSS** command.

Example

The following example deletes the RS secondary server named **ids_stores2**:

EXECUTE FUNCTION task("ha rss delete","ids_stores2");

Related reference:

“onmode -d: Set High Availability server characteristics” on page 14-8

ha sds clear argument: Stop shared-disk replication (SQL administration API)

Use the **ha sds clear** argument with the **admin()** or **task()** function to stop replication to shared disk (SD) secondary servers and convert the primary server to a standard server.

Syntax

►►EXECUTE FUNCTION admin (—"ha sds clear"—,"primary_server"—) ; ►►
task

Element	Description	Key Considerations
<i>primary_server</i>	The name of the primary server to convert to a standard server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function on an established primary server to stop replication to the SD secondary servers.

This function is equivalent to the **onmode -d clear SDS primary** command.

Example

The following example stops replication from the primary server named **ids_stores** to SD secondary servers:

```
EXECUTE FUNCTION task("ha sds clear","ids_stores");
```

Related reference:

“onmode -d: Set High Availability server characteristics” on page 14-8

ha sds primary argument: Convert an SD secondary server to a primary server (SQL administration API)

Use the **ha sds primary** argument with the **admin()** or **task()** function to change a shared disk (SD) secondary server to a primary server.

Syntax

```

▶▶—EXECUTE FUNCTION—admintask—("ha sds primary""ha sds primary force"—  

▶—,—"secondary_server"—)—;—▶▶

```

Element	Description	Key Considerations
<i>secondary_server</i>	The name of the SD secondary server to set as a primary server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function on an established SD secondary server to convert it to the primary server.

Use the **ha sds primary** argument to convert an inactive SD secondary server to a primary server, if the SD secondary servers are connected to it.

Use the **ha sds primary force** argument to convert an inactive SD secondary server to a primary server, whether or not any SD secondary servers are connected to it. If sessions are active, the call succeeds, but the shared disk subsystem can become corrupted.

This function is equivalent to the **onmode -d make primary** command.

Example

The following example converts an SD secondary server named **ids_stores3** to the primary server:

```
EXECUTE FUNCTION task("ha sds primary","ids_stores3");
```

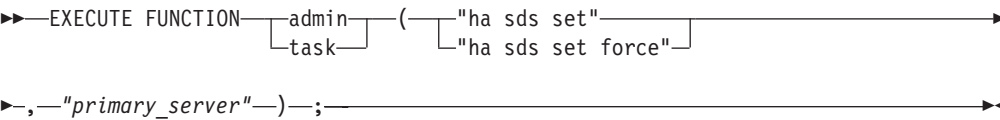
Related reference:

“onmode -d: Set High Availability server characteristics” on page 14-8

ha sds set argument: Create a shared-disk primary server (SQL administration API)

Use the **ha sds set** argument with the **admin()** or **task()** function to define a primary server to replicate to shared disk (SD) secondary servers.

Syntax



Element	Description	Key Considerations
<i>primary_server</i>	The name of the database server to set as a primary server.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function on a standard server to define it as a primary server for SD secondary servers.

Use the **ha sds set** argument to define an inactive standard server as a primary server, if the SD secondary servers are connected to it.

Use the **ha sds set force** argument to define an inactive standard server as a primary server, whether or not any SD secondary servers are connected to it. If sessions are active, the call succeeds, but the shared disk subsystem can become corrupted.

This function is equivalent to the **onmode -d set SDS primary** command.

Example

The following example converts a standard server named `ids_stores` to a primary server:

```
EXECUTE FUNCTION task("ha sds set","ids_stores");
```

Related reference:

“`onmode -d`: Set High Availability server characteristics” on page 14-8

ha set idxauto argument: Replicate indexes to secondary servers (SQL administration API)

Use the **ha set idxauto** argument with the **admin()** or **task()** function to control whether indexes are automatically replicated to secondary servers.

Syntax

```
►►EXECUTE FUNCTION admin ("ha set idxauto off")—; task ("ha set idxauto on")—; ►►
```

Usage

Run this function on an established primary server to enable or disable automatic index replication to secondary servers.

You can run this function on any type of primary server.

This function is equivalent to the **onmode -d idxauto** command.

Example

The following example enables automatic index replication:

```
EXECUTE FUNCTION task("ha set idxauto on");
```

Related reference:

“`onmode -d` command: Replicate an index with data-replication” on page 14-10

ha set ipl argument: Log index builds on the primary server (SQL administration API)

Use the **ha set ipl** argument with the **admin()** or **task()** function to control whether to log index builds on the primary server.

Syntax

```
►►EXECUTE FUNCTION admin ("ha set ipl off")—; task ("ha set ipl on")—; ►►
```

Usage

Run this function on an established primary server to enable or disable the logging of index builds. This function resets the value of the `LOG_INDEX_BUILDS` configuration parameter in the `ONCONFIG` file.

You can run this function on any type of primary server.

This function is equivalent to the **onmode -wf LOG_INDEX_BUILDS** command.

Example

The following example enables the logging of index builds:

```
EXECUTE FUNCTION task("ha set ipl on");
```

Related reference:

“onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

ha set primary argument: Define an HDR primary server (SQL administration API)

Use the **ha set primary** argument with the **admin()** or **task()** function to define a High-Availability Data Replication (HDR) primary server and specify the secondary server.

Syntax

```
►►EXECUTE FUNCTION admin task ("ha set primary"—,"secondary_server"►►  
►►)►;►►
```

Element	Description	Key Considerations
<i>secondary_server</i>	The name of the HDR secondary server to connect to.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function on a standard server to convert it to an HDR primary server and connect to the specified HDR secondary server. If the connection is successful, replication begins.

This function is equivalent to the **onmode -d primary** command.

Example

The following example converts a standard server named **ids_stores** to an HDR primary server:

```
EXECUTE FUNCTION task("ha set primary","ids_stores");
```


Related reference:

“onmode -d: Set data-replication types” on page 14-6

ha set secondary argument: Define an HDR secondary server (SQL administration API)

Use the **ha set secondary** argument with the **admin()** or **task()** function to define a High-Availability Data Replication (HDR) secondary server and specify the primary server.

Syntax

```
►►EXECUTE FUNCTION admin task ("ha set secondary",—"primary_server"►►  
►►) ; ►►
```

Element	Description	Key Considerations
<i>primary_server</i>	The name of the HDR primary server to connect to.	The name must be defined in the DBSERVERNAME or DBSERVERALIASES configuration parameter, or as an Enterprise Replication group name.

Usage

Run this function on a standard database server to convert it to an HDR secondary server, and connect to the specified primary server. If the connection is successful, replication begins.

This function is equivalent to the **onmode -d secondary** command.

Example

The following example converts a standard server to an HDR secondary server, with a primary server named **ids_stores**:

```
EXECUTE FUNCTION task("ha set secondary","ids_stores");
```

Related reference:

“onmode -d: Set data-replication types” on page 14-6

ha set standard argument: Convert an HDR server into a standard server (SQL administration API)

Use the **ha set standard** argument with the **admin()** or **task()** function to convert a High-Availability Data Replication (HDR) primary or secondary server to a standard server.

Syntax

```
►►EXECUTE FUNCTION admin task ("ha set standard") ; ►►
```

Usage

Run this function on an established HDR primary or secondary server to convert it to a standard server. The connection between the primary and secondary servers is dropped and replication stops. The mode of the other server in the HDR pair is not changed.

This function is equivalent to the **onmode -d standard** command.

Example

The following example converts an HDR secondary server to a standard server:

```
EXECUTE FUNCTION task("ha set standard");
```

Related reference:

“onmode -d: Set data-replication types” on page 14-6

ha set timeout argument: Change SD secondary server timeout (SQL administration API)

Use the **ha set timeout** argument with the **admin()** or **task()** function to change the amount of time in seconds that the primary server waits for acknowledgments from shared disk (SD) secondary servers.

Syntax

```
►►EXECUTE FUNCTION admin  
task ("ha set timeout",—"seconds"—);◄◄
```

Element	Description	Key Considerations
<i>seconds</i>	The number of seconds the primary server waits before disconnecting the SD secondary server.	The value must be a positive integer in the range: from 2 to 2 147 483 647

Usage

Run this function on an established shared disk primary server to specify the amount of time in seconds that the primary server waits for a log position acknowledgment to be sent from an SD secondary server. If there is no log position acknowledgment received from the SD secondary server in the specified amount of time, the primary server disconnects from the SD secondary server and continues. After waiting for the specified number of seconds, the primary server starts removing SD secondary servers if page flushing has timed out while waiting for an SD secondary server.

This function resets the value of the SDS_TIMEOUT configuration parameter in the ONCONFIG file.

This function is equivalent to the **onmode -wf SDS_TIMEOUT** command.

The following example sets the timeout period to 5 seconds:

```
EXECUTE FUNCTION task("ha set timeout","5");
```

Related reference:

“onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

```
EXECUTE FUNCTION task("ha set timeout","5");
```

“onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

Use the **message log delete** argument or **file delete** argument with the **admin()** or **task()** function to specify the particular online, ON-Bar activity, or ON-Bar debug message log to delete.

```

▶▶--EXECUTE FUNCTION--┐admin┐──────────────────────────────────────────▶
                      └task┘
▶-(┌"message log delete"┐,─"full_path_for_message_log_file─")─;─▶◀
   └"file delete"┘

```

Examples

The following examples show the arguments that you can use to delete the `/usr/informix/online.log` file:

```
execute function task("file delete", "/usr/informix/online.log");
```

“message log rotate argument: Rotate the message log file (SQL administration API)”

“message log truncate argument: Delete the contents of a message log file (SQL administration API)” on page 21-64

“file status argument: Display the status of a message log file (SQL administration API)” on page 21-51

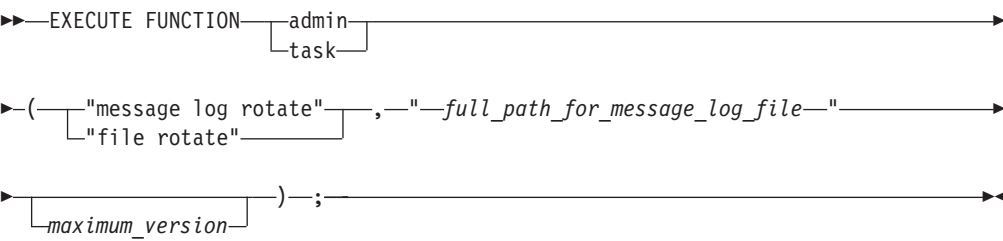
“message log truncate argument: Delete the contents of a message log file (SQL administration API)” on page 21-64

“file status argument: Display the status of a message log file (SQL administration API)” on page 21-51

Use the **message log rotate** argument or the **file rotate** argument with the **admin()** or **task()** function to specify the particular online, ON-Bar activity, or ON-Bar debug message log file to rotate, and to indicate the maximum number of message logs to rotate.

When the message log file rotates, the database server switches to a new online message log file and increments the ID numbers for the previous log files by one. When the maximum number of log files is reached, the log file with the highest ID is deleted.

Syntax



Element	Purpose	Key considerations
<i>full_path_for message_log_file</i>	Full path name of the online, ON-Bar activity, or ON-Bar debug message log file that the server will rotate, for example, /usr/informix/online.log.	
<i>maximum_version</i>	The log file with the highest ID. This is the maximum message log version that the server will rotate.	

Examples

The following examples show the arguments that you can use to rotate a maximum of 52 /usr/informix/online.log files:

```

execute function task("message log rotate", "/usr/informix/online.log",52);
execute function task("file rotate", "/usr/informix/online.log",52);

```

When the database server rotates these files, the server deletes version 52 of the file. Version 51 becomes version 52, version 50 becomes version 51, and so on. The new online log becomes version 1.

- Related reference:**
- “message log truncate argument: Delete the contents of a message log file (SQL administration API)”
 - “message log delete argument: Delete a message log file (SQL administration API)” on page 21-63
 - “file status argument: Display the status of a message log file (SQL administration API)” on page 21-51

message log truncate argument: Delete the contents of a message log file (SQL administration API)

Use the **message log truncate** argument or **file truncate** argument with the **admin()** or **task()** function to specify the particular online, ON-Bar activity, or ON-Bar debug message log file to truncate. When the database server truncates a message log file, it deletes the messages in the log file, but keeps the log file.

Syntax

```

▶▶—EXECUTE FUNCTION—┐admin┐—————▶
                     └task┘

▶—(—┐"message log truncate"—┐,—┐"full_path_for_message_log_file"—┐)—;—▶▶
    └"file truncate"—┘

```

Element	Purpose	Key considerations
<i>full_path_for_message_log_file</i>	Full path name for the online, ON-Bar activity, or ON-Bar debug message log file.	

Examples

The following examples show the arguments that you can use to truncate the `/usr/informix/online.log` file:

```

execute function task("message log truncate", "/usr/informix/online.log");
execute function task("file truncate", "/usr/informix/online.log");

```

Related reference:

“message log rotate argument: Rotate the message log file (SQL administration API)” on page 21-63

“message log delete argument: Delete a message log file (SQL administration API)” on page 21-63

“file status argument: Display the status of a message log file (SQL administration API)” on page 21-51

modify chunk extend argument: Extend the size of a chunk (SQL administration API)

Use the **modify chunk extend** argument with the **admin()** or **task()** function to extend the size of the chunk by a specified minimum amount. The chunk must be marked as extendable.

```

▶▶—EXECUTE FUNCTION—┐admin┐—————▶
                     └task┘

▶—(—┐"modify chunk extend"—┐,—┐"chunk_number"—┐,—┐"extend_amount"—┐)—;—▶▶
    └"chunk_number"—┘└"extend_amount"—┘

```

Element	Description	Key Considerations
<i>chunk_number</i>	The number of the chunk.	
<i>extend_amount</i>	The minimum amount of space in kilobytes to add to the chunk.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

You must mark a chunk as extendable before the chunk can be extended, either manually or automatically. Use the **modify chunk extendable** argument with the **admin()** or **task()** function to mark a chunk as extendable.

The **modify chunk extend** SQL administration API command is an alternative to the **adm_add_storage** task that the server can run to automatically extend the size of a chunk when the space containing the chunk runs low or out of free pages.

You cannot extend a chunk in a mirrored space, and you will receive an error if you provide the number of a mirror chunk when you run a **modify chunk extend** SQL administration API command.

To identify primary and mirror chunks in a mirrored space, look for the P (primary) or M (mirror) in position 1 of the flags field in **onstat -d** command output.

The server might round up the requested size, depending on the page size and the configured create size and extend size of the space.

Examples

Suppose that your **onstat -d** command output shows that chunk number 3 is a mirror chunk and chunk number 4 is not a mirror chunk. You cannot extend the size of chunk number 3. However, you can modify chunk number 4. The following command extends the size of chunk number 4 by 10000 kilobytes:

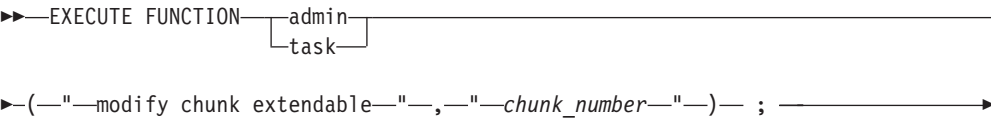
```
EXECUTE FUNCTION task("modify chunk extend", "4", "10000");
```

Related reference:

- “modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)”
- “modify chunk extend argument: Extend the size of a chunk (SQL administration API)” on page 21-65
- “modify space sp_sizes argument: Modify the create or extend size of a storage space (SQL administration API)” on page 21-69
- “modify space expand argument: Expand the size of a space (SQL administration API)” on page 21-68
- “onstat -d command: Print chunk information” on page 20-34

modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)

Use the **modify chunk extendable** argument with the **admin()** or **task()** function to specify that a particular chunk in an unmirrored dbspace or temporary dbspace can be extended..



Element	Description	Key Considerations
chunk_number	The number of the chunk.	

Usage

If a chunk is marked as extendable, either:

- The server can automatically extend the chunk when the unmirrored dbspace or temporary dbspace containing the chunk runs low or out of free pages.
- You can use the **modify chunk extend** argument with the **admin()** or **task()** function to extend the size of the chunk.

However, if the extend size for the dbspace or temporary dbspace is set to 0, the server cannot automatically extend an extendable chunk in that space. In this situation, you can still manually extend the chunk.

The server will automatically mark chunks that are allocated from extendable storage pool entries as extendable. Therefore, you do not need to mark these chunks as extendable. For information on extendable storage pool entries, see “storagepool add argument: Add a storage pool entry (SQL administration API)” on page 21-119.

Chunks in mirrored spaces cannot be extended. If you try to make a mirror chunk extendable, you will receive an error.

To identify primary and mirror chunks in a mirrored space, look for the P (primary) or M (mirror) in position 1 of the flags field in **onstat -d** command output.

Example

The following snippet of **onstat -d** output shows that chunk number 3 is a mirror chunk:

Chunks address	chunk/dbs	offset	size	free	bpages	flags	pathname
451191c8	1	1	0	225000	101572	P0-B--	/reg1/rootchunk
451197d0	2	2	0	1250	1149	P0-B--	/reg1/dbs1
451199d0	3	3	0	1250	1149	P0-B--	/reg1/dbs2
46a36638	3	3	0	1250	0	M0-B--	/reg1/chunk2
45119bd0	4	4	0	1250	1149	P0-B--	/reg1/dbs3

Thus, you cannot extend the size of chunk number 3. However, you can specify that chunk number 4 is extendable, as follows:

```
EXECUTE FUNCTION sysadmin:task("modify chunk extendable", "4");
```

Related reference:

“modify chunk extendable off argument: Mark a chunk as not extendable (SQL administration API)”

“modify space sp_sizes argument: Modify the create or extend size of a storage space (SQL administration API)” on page 21-69

“modify chunk extend argument: Extend the size of a chunk (SQL administration API)” on page 21-65

“modify space expand argument: Expand the size of a space (SQL administration API)” on page 21-68

modify chunk extendable off argument: Mark a chunk as not extendable (SQL administration API)

Use the **modify chunk extendable off** argument with the **admin()** or **task()** function to specify that a particular chunk cannot be extended.

```

>>EXECUTE FUNCTION admin
>>task
>>

>("—modify chunk extendable off—",—"—chunk number—")— ; —————>>

```

Element	Description	Key Considerations
<i>chunk_number</i>	The number of the chunk.	

Usage

The default status for chunks is not extendable. If you previously marked a chunk as extendable, you can change the status to not extendable.

If a chunk is marked as not extendable:

- The server cannot automatically extend the chunk when the space containing the chunk runs low or out of free pages.
- You cannot manually extend the size of the chunk.

If the storage pool contains entries, the server can extend a storage space by adding another chunk to the storage space.

Example

The following example specifies that the you or the server cannot extend chunk 9:

```
EXECUTE FUNCTION task("modify chunk extendable off", "9");
```

Related reference:

“modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)” on page 21-66

modify space expand argument: Expand the size of a space (SQL administration API)

Use the **modify space expand** argument with the **admin()** or **task()** function to immediately expand the size of a space, when you do not want to wait for Informix to automatically expand the space.

```

▶▶EXECUTE FUNCTION admin(—"modify space expand—"
└─task┘
▶,—"space name"—,"size"—);
▶▶

```

Element	Description	Key Considerations
<i>space_name</i>	The name of the storage space.	
<i>size</i>	The minimum size by which you want to expand the space.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

The **modify space expand** SQL administration API command expands a storage space immediately, either by extending an extendable chunk in the space or by adding a new chunk. The create size and extend size settings for the space do not affect this operation.

The actual number of kilobytes added to the space might exceed your requested size, depending on factors such as the page size of the space and the chunk size settings for available entries in the storage pool.

The storage pool must contain entries (such as raw devices, cooked files, or directories) that the server can use to expand the space.

After you run a **modify space expand** SQL administration API command, Informix first attempts to expand the space by extending an extendable chunk in the space. If the space does not contain any extendable chunks, the server uses entries in the storage pool to expand the space.

You cannot expand a mirrored storage space.

Examples

The following command expands dbspace5 by 10 megabytes:

```
EXECUTE FUNCTION task("modify space expand", "dbspace5", "10 MB");
```

Related reference:

“modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)” on page 21-66

“modify chunk extend argument: Extend the size of a chunk (SQL administration API)” on page 21-65

“modify space sp_sizes argument: Modify the create or extend size of a storage space (SQL administration API)”

“create chunk from storagepool argument: Create a chunk from the storage pool (SQL administration API)” on page 21-30

modify space sp_sizes argument: Modify the create or extend size of a storage space (SQL administration API)

Use the **modify space sp_sizes** argument with the **admin()** or **task()** function to modify two different sizes that are associated with expanding a storage space, the extend size and the create size. Do this to control how Informix uses storage pool entries for a particular storage space.

```
►► EXECUTE FUNCTION 

|       |
|-------|
| admin |
| task  |

 ( 

|  |
|--|
|  |
|--|

 )
```

```
► "—" modify space sp_sizes—"—,—"space_name"—" 

|  |
|--|
|  |
|--|


```

```
►,—"new_create_size"—,—"new_extend_size"—) —; 

|  |
|--|
|  |
|--|


```

Element	Description	Key Considerations
<i>space_name</i>	The name of the storage space.	

Element	Description	Key Considerations
<i>new_create_size</i>	The minimum size of a new chunk that the server can create when automatically expanding this space using the storage pool. You can define the size as a number of kilobytes or as a percentage of the total space.	<p>The default create size is set to 10 percent of the total size of the space.</p> <p>The size you specify affects chunks the server creates automatically. It does not affect any manual chunks that you might create for the associated space.</p>
<i>new_extend_size</i>	The minimum size that the server can use when automatically extending a chunk in an unmirrored dbspace or temporary dbspace. The size can be a specified number of kilobytes or a percentage of the total space.	<p>The default extend size is 10 megabytes.</p> <p>The size you specify affects chunks the server extends automatically. It does not affect any manual chunk extensions that you might initiate for the associated space.</p>

Usage

If the create or extend size value is 100 or a lower value, Informix interprets the value as a percentage (for example, 10 = 10 percent and 2.84 = 2.84 percent). If the value is 1000 or higher, the server interprets the value as a specific number of kilobytes. Values between 100 and 1000 are not valid.

If you set the create size and the extend size to 0, Informix does not automatically expand the space, even when the space becomes full. Additionally, if you set the extend size to 0, you also remove the "Extendable" flag from all chunks in that space. This is an easy way to mark all chunks in a space as not extendable, using one operation.

The create and extend size values are minimum sizes. The actual size by which a space is expanded might be larger, depending on the chunk size of the storage pool entry that the server is using or the amount of space that the server needs at that particular time.

For example, suppose you created a storage pool entry to expand storage space when necessary. Then suppose that a dbspace named "logdbs" is out of free pages and requires an additional 500 megabytes for a new log. If none of the chunks in logdbs can be extended, Informix adds a new chunk that has the minimum size specified by the create size value for the logdbs dbspace. If the create size for the logdbs dbspace is less than or equal to 500 megabytes, the server attempts to find a minimum of 500 megabytes of space. If the create size for logdbs is 1 gigabyte, the server ignores the requested size and adds a 1 gigabyte chunk.

If the server is unable to find the minimum amount of space required at any given point, the server returns an out-of-space error and the log creation fails.

Examples

The following command sets the minimum create size to 60 megabytes and the minimum extend size to 10 megabytes for dbspace3:

```
EXECUTE FUNCTION task("modify space sp_sizes", "dbspace3", "60000", "10000");
```

The following command sets the minimum create size to 20 percent and the minimum extend size 1.5 percent for dbspace8:

```
EXECUTE FUNCTION task("modify space sp_sizes", "dbspace8", "20", "1.5");
```

Related reference:

“modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)” on page 21-66

“modify chunk extend argument: Extend the size of a chunk (SQL administration API)” on page 21-65

“modify space expand argument: Expand the size of a space (SQL administration API)” on page 21-68

onmode and a arguments: Add a shared-memory segment (SQL administration API)

Use the **onmode** and **a** arguments with the **admin()** or **task()** function to add a shared-memory segment.

Syntax

```
►►EXECUTE FUNCTION admin task (—"onmode"—,—"a"— , —"size"—);►►
```

Element	Description	Key Considerations
<i>size</i>	The size, in kilobytes, of a new virtual shared-memory segment.	The value of <i>size</i> must be a positive integer that does not exceed the operating-system limit on the size of shared-memory segments.

Usage

Ordinarily, you do not need to add segments to the virtual portion of shared memory because the database server automatically adds segments as they are needed. However, as segments are added, the database server might reach the operating-system limit for the maximum number of segments before it acquires the memory that it needs. This situation typically occurs when the SHMADD configuration parameter is set so small that the database server exhausts the number of available segments before it acquires the memory that it needs for some operation.

You can use this function to add a segment that is larger than the size specified by the SHMADD configuration parameter. By using this function to add a segment, you can adhere to the operating system limit for segments while meeting the need that the database server has for more memory.

This function is equivalent to the **onmode -a** command.

Example

The following example adds 500 KB of virtual shared-memory:

```
EXECUTE FUNCTION task("onmode","a","500");
```

Related reference:

"onmode -a: Add a shared-memory segment" on page 14-2

onmode and c arguments: Force a checkpoint (SQL administration API)

Use the **onmode** and **c** arguments with the **admin()** or **task()** function to force a checkpoint.

Syntax

```
►► EXECUTE FUNCTION admin task (—"onmode"—,—"c"—, "hard"  
"block"  
"norm"  
"unblock")—; ►►
```

Usage

This function forces a checkpoint that flushes the buffers to disk. You can use the **c** option to force a checkpoint if the most recent checkpoint record in the logical log was preventing the logical-log file from being freed (status U-B-L).

Use the **block** argument to prevent the database server from processing any transactions. Use this option to perform an external backup on Informix. While the database server is blocked, users cannot access it, except in read-only mode. No transactions can complete until the database server is unblocked.

Use the **hard** argument to force a blocking checkpoint. This is the default.

Use the **norm** argument to force a nonblocking checkpoint.

Use the **unblock** argument to unblock the database server. When the database server is unblocked, data transactions and normal database server operations can resume. Use this option after you complete an external backup on Informix.

This function is equivalent to the **onmode -c** command.

Example

The following example starts a blocking checkpoint:

```
EXECUTE FUNCTION task("onmode","c","hard");
```

“onmode -c: Force a checkpoint” on page 14-4

Use the **onmode** and **C** arguments with the **admin()** or **task()** function to control the B-tree scanner for cleaning indexes of deleted items.

```

▶▶EXECUTE FUNCTION admin (—"onmode"—,"C"—,
└─task┘
)

"start"—,"1"—
└─"start_count"—┘
) ;

"stop"—,"1"—
└─"stop_count"—┘

"threshold"—,"thresh_size"—
"duration"—,"dur_num"—
"rangesize"—,"range_size"—
"alice"—,"alice_mode"—
"compression"—,"compression_level"—

```

Element	Description	Key Considerations
<i>alice_mode</i>	The alice mode for the system.	Valid integer values range from 0 (OFF) to 12.
<i>compression_level</i>	For a database server instance, the level at which two partially used index pages are merged. The pages are merged if the data on those pages totals a set level.	Valid values for the level are low, med (medium), high, and default. The system default value is med.
<i>dur_num</i>	The number of seconds that the hot list is valid.	After this number of seconds expires, the hot list will be rebuilt by the next available B-tree scanner thread, even if unprocessed items are on the list. Scanners that are processing requests are not interrupted.
<i>range_size</i>	The size of an index before index range cleaning is enabled.	A size of -1 can be used to disable range scanning.
<i>start_count</i>	The number of B-tree scanner threads to start.	If <i>start_count</i> is not specified, 1 more thread is started. A maximum of 32 threads can be started at one time. But, there is no limit on the number of scanner threads run simultaneously.

Element	Description	Key Considerations
<i>stop_count</i>	The number of B-tree scanner threads to stop.	<p>If <i>stop_count</i> is not specified, a single thread is stopped. Stopping all index scanners prevents all index cleaning.</p> <p>If you specify a larger <i>stop_count</i> value than the number of threads that are running, no error is issued, but all scanner threads are stopped.</p>
<i>thresh_size</i>	The minimum number of deleted items an index must encounter before an index is placed on the hot list.	After all indexes above the threshold have been cleaned and there is no other work for the B-tree scanner to do, the indexes below the threshold are added to the hot list.

Usage

The B-tree scanner has statistical information that tracks index efficiency and how much extra work the index places on the server. Based on the amount of extra work the index has accomplished because of committed deleted index items, the B-tree scanner develops an ordered list of indexes that have caused the server to do extra work, called the hot list. The index causing the highest amount of extra work is cleaned first and the rest of the indexes are cleaned in descending order. The DBA can allocate cleaning threads dynamically to configure workloads.

This function is equivalent to the **onmode -C** command.

Example

The following commands start 60 B-tree scanner threads:

```
EXECUTE FUNCTION admin("onmode","C","start","30");
EXECUTE FUNCTION admin("onmode","C","start","30");
```

The following command stops all of these threads:

```
EXECUTE FUNCTION admin("onmode","C","stop","30000");
```

No error is issued when the *stop_count* value is greater than the number of running threads.

Related reference:

“onmode -C: Control the B-tree scanner” on page 14-5

onmode and d arguments: Set data-replication types (SQL administration API)

Use the **onmode** and **d** arguments with the **admin()** or **task()** function to change the mode of a server participating in high-availability data replication (HDR).

Syntax

```
►► EXECUTE FUNCTION admin  
task ("onmode"—,—"d"—,—————►
```



Element	Description	Key Considerations
<i>dbserver</i>	The name of the primary or secondary database server.	The <i>dbserver</i> name must correspond to the name specified in the DBSERVERNAME configuration parameter in the ONCONFIG file of the intended secondary database server. The name should <i>not</i> correspond to one of the database servers that the DBSERVERALIASES configuration parameter specifies.

Usage

Use this function to set the High-Availability Data Replication type as standard, primary, or secondary. You can use the **primary** and **secondary** arguments only when the database server is in quiescent mode. You can use the **standard** argument when the database server is in quiescent, online, or read-only mode.

The *dbserver* argument of the other database server in the data-replication pair and the type of a database server (standard, primary, or secondary) is preserved after reinitialization of shared memory.

The **standard** argument drops the connection between database servers in a data replication pair (if one exists) and sets the database server type of the current database server to standard. This option does not change the mode or type of the other database server in the pair.

The **primary** and *dbserver* arguments set the database server type to primary and attempt to connect with the database server that *dbserver* specifies. If the connection is successful, data replication is turned on. The primary database server goes into online mode, and the secondary database server goes into read-only mode. If the connection is not successful, the database server comes to online mode, but data replication is not turned on.

The **secondary** and *dbserver* arguments set the database server type to secondary and attempt to connect with the database server that *dbserver* specifies. If the connection is successful, data replication is turned on. The primary database server goes online, and the secondary database server goes into read-only mode. If the connection is not successful, the database server comes to read-only mode, but data replication is not turned on.

This function is equivalent to the **onmode -d** command.

Example

The following example sets a server named **ids_stores** as an HDR primary server:

```
EXECUTE FUNCTION task("onmode","d","primary","ids stores");
```

Related reference:

“onmode -d: Set data-replication types” on page 14-6

onmode and D arguments: Set PDQ priority (SQL administration API)

Use the **onmode** and **D** arguments with the **admin()** or **task()** function to temporarily reset the PDQ resources that the database server can allocate to any one decision-support query.

Syntax

```
►►EXECUTE FUNCTION admin  
task (—"onmode"— , —"D"—,—"max_priority"—►  
  
►) —; ►►
```

Element	Description	Key Considerations
<i>max_priority</i>	The percentage of the user-requested PDQ resources actually allocated to the query.	The value must be an unsigned integer from 0 to 100.

Usage

Use this function to override the limit set by the MAX_PDQPRIORITY configuration parameter while the database server is online. The new values affect only the current instance of the database server; the values are not recorded in the **onconfig** file. If you shut down and restart the database server, the values of the parameter reverts to the values in the **onconfig** file.

This function is equivalent to the **onmode -D** command.

Example

The following example sets the percentage of PDQ resources that can be allocated to a query to 50 percent:

```
EXECUTE FUNCTION task("onmode","D","50");
```

Related reference:

“onmode -D, -M, -Q, -S: Change decision-support parameters” on page 14-11

onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)

Use the **onmode** and **e** arguments with the **admin()** or **task()** function to temporarily change the mode of the SQL statement cache.

Syntax

```
►►EXECUTE FUNCTION admin  
task (—"onmode"—,—"e"—, "enable"  
"flush"  
"off"  
"on") —; ►►
```


Usage

Use the **enable** argument to enable the SQL statement cache if it is disabled. Individual user sessions can use the statement cache only after they perform either of the following actions:

- Set the environment variable **STMT_CACHE** to 1.
- Execute the SQL statement **SET STATEMENT CACHE ON**.

Use the **flush** argument to flush the statements that are not in use from the SQL statement cache, which remains enabled. After the cache is flushed, the **onstat -g ssc ref_cnt** field shows 0.

Use the **off** argument to turn off the SQL statement cache, so that no statements are cached.

Use the **on** argument to cache all statements except those a user turns off by one of the following actions:

- Use this command to specify the OFF mode.
- Set the environment variable **STMT_CACHE** to 0.
- Execute the SQL statement **SET STATEMENT CACHE OFF** statement.

This function cannot modify the **STMT_CACHE** configuration parameter setting in the **ONCONFIG** file, but the last argument overrides that setting (or the default value, if **STMT_CACHE** is not set). Any changes to the statement cache behavior that you make with this command are in effect for the current database server session only. When you restart the database server, it uses the setting of the **STMT_CACHE** parameter in the **ONCONFIG** file. If the **STMT_CACHE** configuration parameter is not defined in the **ONCONFIG** file, the server does not use a statement cache.

This function is equivalent to the **onmode -e** command.

Example

The following example enables the SQL statement cache:

```
EXECUTE FUNCTION task("onmode","e","enable");
```

Related reference:

“onmode -e: Change usage of the SQL statement cache” on page 14-13

onmode and F arguments: Free unused memory segments (SQL administration API)

Use the **onmode** and **F** arguments with the **admin()** or **task()** function to free unused memory segments.

Syntax

```
➤➤ EXECUTE FUNCTION admin task ("onmode",—"F"—); ➤➤
```

Usage

When you execute this function, the memory manager examines each memory pool for unused memory. The memory manager immediately frees unused blocks of memory that it locates. After the memory manager checks each memory pool, it begins checking memory segments and frees any that the database server no longer needs.

Running this command causes a significant degradation of performance for any users that are active when you execute the utility. Although the execution time is brief (1 to 2 seconds), degradation for a single-user database server can reach 100 percent. Systems with multiple CPU virtual processors experience proportionately less degradation.

To confirm that the unused memory was freed, check the message log. If the memory manager frees one or more segments, it displays a message that indicates how many segments and bytes of memory were freed.

Tip: Run this command from an operating-system scheduling facility regularly and after the database server performs any function that creates more memory segments, including large index builds, sorts, or backups.

This function is equivalent to the **onmode -F** command.

Example

The following example frees unused memory blocks:

```
EXECUTE FUNCTION task("onmode","F");
```

Related reference:

“onmode -F: Free unused memory segments” on page 14-13

onmode and j arguments: Switch the database server to administration mode (SQL administration API)

Use the **onmode** and **j** arguments with the **admin()** or **task()** function to change the database server to administration mode.

Syntax

```
►►—EXECUTE FUNCTION—

|       |
|-------|
| admin |
| task  |

—(—"onmode"—,—"j"—)—;—►◄
```

Usage

When the server is changed to administration mode, all sessions lose their connection to the database server except for sessions of the following users:

- User **informix**
- Users in the **DBSA** group
- Users who are identified in **ADMIN_MODE_USERS** settings

This function is equivalent to the **onmode -j** command.

Example

The following example changes the server to administration mode:

```
EXECUTE FUNCTION task("onmode","j");
```

Related reference:

“onmode -k, -m, -s, -u, -j: Change database server mode” on page 14-15

onmode and l arguments: Switch to the next logical log (SQL administration API)

Use the **onmode** and **l** arguments with the **admin()** or **task()** function to switch the current logical-log file to the next logical-log file.

Syntax

```
►►—EXECUTE FUNCTION—

|       |
|-------|
| admin |
| task  |

—(—"onmode"—,—"l"—)—;—►►
```

Usage

This function is equivalent to the **onmode -l** command.

For information on switching to the next logical-log file, see the section on managing logical-log files in the *IBM Informix Administrator's Guide*.

Example

The following example moves the logical log out of the **root** chunk

```
SELECT task("onmode", "l") FROM sysmaster:syslogfil  
WHERE chunk = 1 AND sysmaster:bitval(flags,"0x02")>0;
```

Related reference:

“onmode -l: Switch the logical-log file” on page 14-18

onmode and m arguments: Switch to multi-user mode (SQL administration API)

Use the **onmode** and **m** arguments with the **admin()** or **task()** function to change the database server to multi-user mode.

Syntax

```
►►—EXECUTE FUNCTION—

|       |
|-------|
| admin |
| task  |

—(—"onmode"—,—"m"—)—;—►►
```

Usage

Use this function to bring the database server online from quiescent mode or from administration mode.

This function is equivalent to the **onmode -m** command.

Example

The following example changes the server to multi-user mode:

```
EXECUTE FUNCTION task("onmode","m");
```

Related reference:

“onmode -k, -m, -s, -u, -j: Change database server mode” on page 14-15

onmode and M arguments: Temporarily change decision-support memory (SQL administration API)

Use the **onmode** and **M** arguments with the **admin()** or **task()** function to temporarily change the size of memory available for parallel queries.

Syntax

►►EXECUTE FUNCTION admin task ("onmode"—,"M"—,"size"—);◄◄

Element	Description	Key Considerations
<i>size</i>	The new size limit (in kilobytes) of the maximum amount of memory available for parallel queries.	The maximum value for 32-bit platform is 2 gigabytes. The maximum value for 64-bit platform is 4 gigabytes.

Usage

Use this function to override the limit set by the DS_TOTAL_MEMORY configuration parameter while the database server is online. The new values affect only the current instance of the database server; the values are not recorded in the ONCONFIG file. If you shut down and restart the database server, the values of the parameter revert to the values in the ONCONFIG file.

This function is equivalent to the **onmode -M** command.

Example

The following example sets the size limit for parallel queries to 50 MB:

```
EXECUTE FUNCTION task("onmode","M","50000");
```

Related reference:

“onmode -D, -M, -Q, -S: Change decision-support parameters” on page 14-11

onmode and n arguments: Unlock resident memory (SQL administration API)

Use the **onmode** and **n** arguments with the **admin()** or **task()** function to end forced residency of the resident portion of shared memory.

Syntax

```
►►—EXECUTE FUNCTION admin task ("onmode", "n");
```

Usage

The **RESIDENT** configuration parameter must be set to 1 in the **ONCONFIG** file before you can run this function.

This function does not affect the value of the **RESIDENT** configuration parameter, the forced-residency parameter in the **ONCONFIG** file.

This function is equivalent to the **onmode -n** command.

Example

The following example unlocks resident memory:

```
EXECUTE FUNCTION task("onmode", "n");
```

Related reference:

“onmode -n, -r: Change shared-memory residency” on page 14-18

onmode and 0 arguments: Mark a disabled dbspace as down (SQL administration API)

Use the **onmode** and **0** arguments with the **admin()** or **task()** function to mark a disabled dbspace as down so that the checkpoint that is being blocked by the disabled dbspace can continue and any blocked threads are released.

Syntax

```
►►—EXECUTE FUNCTION admin task ("onmode", "0");
```

Usage

This function overrides the **WAIT** mode of the **ONDBSPACEDOWN** configuration parameter. Use this command only in the following circumstances:

- **ONDBSPACEDOWN** is set to **WAIT**.
- A disabling I/O error occurs that causes the database server to block all updating threads.

- You cannot or do not want to correct the problem that caused the disabling I/O error.
- You want the database server to mark the disabled dbspace as down and continue processing.

This function is equivalent to the **onmode -O** command.

Example

The following example marks disabled dbspaces as down:

```
EXECUTE FUNCTION task("onmode","0");
```

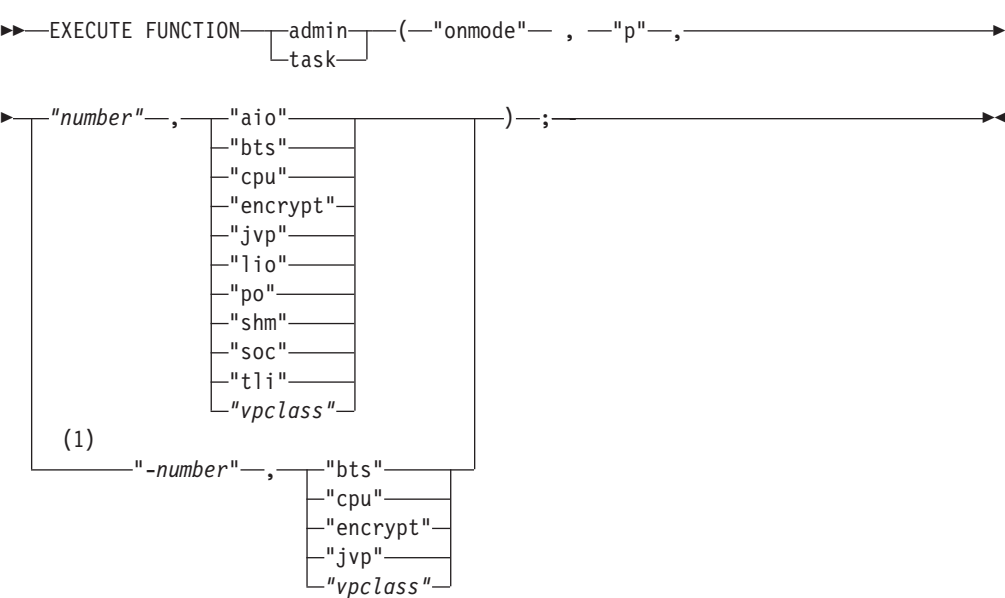
Related reference:

“onmode -O: Override ONDBSPACEDOWN WAIT mode” on page 14-19

onmode and p arguments: Add or remove virtual processors (SQL administration API)

Use the **onmode** and **p** arguments with the **admin()** or **task()** function to dynamically add or remove virtual processors for the current database server session. This function does not update the onconfig file.

Syntax



Notes:

- 1 UNIX only

Element	Description	Key Considerations
<i>number</i>	The number of virtual processors to add or to remove.	<p>A positive number adds virtual processors. The maximum number of virtual processors you can add depends on the operating system.</p> <p>UNIX: A negative number removes virtual processors. The number of virtual processors to drop cannot exceed the actual number of processors of the specified type.</p>

Element	Description	Key Considerations
<i>vpclass</i>	The name of a user-defined virtual processor class.	Windows: The <i>number</i> argument must be set to 1 because you can only create one instance of a user-defined virtual processor.

Usage

You can use this function only when the database server is in online mode.

The number of CPU VPs should not exceed the number of physical processors on your system, but no error is issued if they do. The database server uses the number of CPU VPs to allocate resources for parallel database queries (PDQ). If you drop CPU VPs, your queries might run significantly slower. After you change the number of CPU VPs, the **Reinit** field in the output from the **onstat -g mgm** command shows how many queries are waiting for other queries to complete.

See the *IBM Informix Performance Guide* for more information about performance implications of the CPU VP class.

For a description of each virtual processor class, see the *IBM Informix Administrator's Guide*.

This function is equivalent to the **onmode -p** command.

Example

The following example adds one CPU virtual processor:

```
EXECUTE FUNCTION task("onmode","p","1","cpu");
```

The following example removes one Java virtual processor:

```
EXECUTE FUNCTION task("onmode","p","-1","jvp");
```

Related reference:

“onmode -p: Add or remove virtual processors” on page 14-19

onmode and Q arguments: Set maximum number for decision-support queries (SQL administration API)

Use the **onmode** and **Q** arguments with the **admin()** or **task()** function to change the maximum number of concurrently executing decision-support queries.

Syntax

```
►►—EXECUTE FUNCTION admin  
task (—"onmode"—,—"Q"—,—"queries"—)—;—►►
```

Element	Description	Key Considerations
<i>queries</i>	The maximum number of concurrently executing parallel queries.	The number must be an unsigned integer from 1 to 8,388,608.

Usage

Use this function to override the limit set by the DS_MAX_QUERIES configuration parameter while the database server is online. The new values affect only the current instance of the database server; the values are not recorded in the ONCONFIG file. If you shut down and restart the database server, the values of the parameter revert to the values in the ONCONFIG file.

For information on parameters used for controlling PDQ, see the *IBM Informix Performance Guide*.

This function is equivalent to the **onmode -Q** command.

Example

The following example sets the maximum number of concurrently executing parallel queries to 8:

```
EXECUTE FUNCTION task("onmode","Q","8");
```

Related reference:

“onmode -D, -M, -Q, -S: Change decision-support parameters” on page 14-11

onmode and r arguments: Force residency of shared memory (SQL administration API)

Use the **onmode** and **r** arguments with the **admin()** or **task()** function to start forced residency of the resident portion of shared memory.

Syntax

```
►►—EXECUTE FUNCTION admin  
task (—"onmode"—,—"r"—);—————►►
```

Usage

The RESIDENT configuration parameter must be set to 1 in the ONCONFIG file before you can run this function.

This function does not affect the value of the RESIDENT configuration parameter, the forced-memory parameter in the ONCONFIG file.

This function is equivalent to the **onmode -r** command.

Example

The following example starts forced residency of shared memory:

```
EXECUTE FUNCTION task("onmode","r");
```


Related reference:

"onmode -n, -r: Change shared-memory residency" on page 14-18

onmode and S arguments: Set maximum number of decision-support scans (SQL administration API)

Use the **onmode** and **S** arguments with the **admin()** or **task()** function to change the maximum number of concurrently executing decision-support scans for the current session.

Syntax

```
►►—EXECUTE FUNCTION admin task ("onmode",—"S"—,"scans"—);—►►
```

Element	Description	Key Considerations
<i>scans</i>	The maximum number of concurrently executing parallel scans.	The number must be an unsigned integer from 10 to 1 048 576.

Usage

Use this function to override the limit set by the DS_MAX_SCANS configuration parameter while the database server is online. The new value affects only the current instance of the database server; the values are not recorded in the ONCONFIG file. If you shut down and restart the database server, the value of the parameter reverts to the value in the ONCONFIG file.

For information on parameters used for controlling PDQ, see the *IBM Informix Performance Guide*.

This function is equivalent to the **onmode -S** command.

Example

The following example sets the maximum number of concurrently executing parallel scans to 2000:

```
EXECUTE FUNCTION task("onmode","S","2000");
```

Related reference:

"onmode -D, -M, -Q, -S: Change decision-support parameters" on page 14-11

onmode and W arguments: Reset statement cache attributes (SQL administration API)

Use the **onmode** and **W** arguments with the **admin()** or **task()** function to change whether and when a statement can be inserted into the SQL cache.

Syntax

```
►►—EXECUTE FUNCTION admin task ("onmode",—"W"—, —►►
```

```

    ("STMT_CACHE_HITS"—,—"hits"—)——;
    ("STMT_CACHE_NOLIMIT"—,—"value"—)——;

```

Element	Description	Key Considerations
<i>hits</i>	The number of hits (references) to a statement before it is fully inserted in the SQL statement cache.	Possible values are: <ul style="list-style-type: none"> • 0 = Insert all qualified statements and their memory structures in the cache. • 1 or more = Exclude ad hoc queries from entering the cache.
<i>value</i>	Whether statements are inserted in the SQL statement cache.	Possible values are: <ul style="list-style-type: none"> • 0 = The database server does not insert statements into the cache. • 1 = The database server always inserts statements in the cache.

Usage

Use this function to reset the value of the STMT_CACHE_HITS or STMT_CACHE_NOLIMIT configuration parameter while the database server is online. The new value affects only the current instance of the database server; the value is not recorded in the ONCONFIG file. If you shut down and restart the database server, the value of the parameter reverts to the value in the ONCONFIG file.

If you set the value of STMT_CACHE_HITS equal to 0, the database server inserts all qualified statements and their memory structures in the cache. If the value is greater than 0 and the number of times the SQL statement has been executed is less than the value of STMT_CACHE_HITS, the database server inserts *key-only* entries in the cache. The database server inserts qualified statements in the cache after the specified number of hits has occurred for the statement. The new value of STMT_CACHE_HITS displays in the **#hits** field of the **onstat -g ssc** output.

If none of the queries are shared, set STMT_CACHE_NOLIMIT to 0 to prevent the database server from allocating a large amount of memory for the statement cache.

This function is equivalent to the **onmode -W** command.

Example

The following example prevents ad hoc queries from entering the SQL statement cache:

```
EXECUTE FUNCTION task("onmode","W","STMT_CACHE_HITS","1");
```

Related reference:

“onmode -W: Change settings for the SQL statement cache” on page 14-24

onmode and wf arguments: Permanently update a configuration parameter (SQL administration API)

Use the **onmode** and **wf** arguments with the **admin()** or **task()** function to dynamically update the value of a configuration parameter in the **onconfig** file.

Syntax

```
►►EXECUTE FUNCTION admin  
task (—————►  
  
►—"onmode"—,—"wf"—,—"configuration_parameter=value"—)——►
```

Element	Description	Key Considerations
<i>configuration_parameter</i>	The name of a configuration parameter.	The configuration parameter must be one that you can update dynamically. The list of configuration parameters that you can update dynamically is the same as for the onmode -wf command.
<i>value</i>	The new value or values for the configuration parameter.	The value must be valid for the configuration parameter. The format of the new value must conform exactly to the syntax for that configuration parameter.

Usage

Use this function to permanently update the value of a configuration parameter. The new value takes effect immediately and persists in the ONCONFIG file after the server restarts.

This function is equivalent to the **onmode -wf** command.

Example

The following example sets the value of the DYNAMIC_LOGS configuration parameter to 2 in the **onconfig** file:

```
EXECUTE FUNCTION task("onmode","wf","DYNAMIC_LOGS=2");
```




Related reference:

"onmode -wf, -wm: Dynamically change certain configuration parameters" on page 14-25

onmode and wm arguments: Temporarily update a configuration parameter (SQL administration API)

Use the **onmode** and **wm** arguments with the **admin()** or **task()** function to dynamically update the value of a configuration parameter in memory.

Syntax

►► EXECUTE FUNCTION  ()
►► "onmode"—,"wm"—,"configuration_parameter=value"—) ; 

Element	Description	Key Considerations
<i>configuration_parameter</i>	The name of a configuration parameter.	<p>The configuration parameter that you specify must be one that you can update dynamically.</p> <p>The list of configuration parameters that you can update dynamically is the same as for the onmode -wf command.</p>
<i>value</i>	The new value or values for the configuration parameter.	<p>The value must be valid for the configuration parameter.</p> <p>The format of the new value must conform exactly to the syntax for that configuration parameter.</p>

Usage

Use this function to temporarily update the value of a configuration parameter that can be dynamically updated. The new value takes effect immediately. The new value is not written to the ONCONFIG file and is lost when the database server is restarted.

This function is equivalent to the **onmode -wm** command.

Example

The following example sets the value of the DYNAMIC_LOGS configuration parameter to 2 for the current session:

```
EXECUTE FUNCTION task("onmode","wm","DYNAMIC_LOGS=2");
```

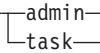

Related reference:

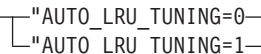
"onmode -wf, -wm: Dynamically change certain configuration parameters" on page 14-25

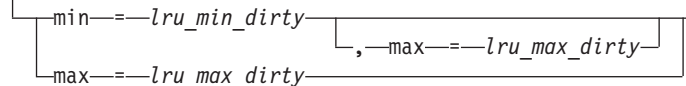
onmode, wm, and AUTO_LRU_TUNING arguments: Change LRU tuning status (SQL administration API)

Use the **onmode**, **wm**, and **AUTO_LRU_TUNING** arguments with the **admin()** or **task()** function to dynamically update LRU tuning status and the flushing behavior for the current session.

Syntax

```
►► EXECUTE FUNCTION  ("onmode",—"wm"—, )

►  "AUTO_LRU_TUNING=0"
  "AUTO_LRU_TUNING=1"

►  min—=lru_min_dirty
  max—=lru_max_dirty
  , max—=lru_max_dirty
  "—)" ;
```

Element	Description	Key Considerations
<i>lru_max_dirty</i>	The percentage of modified pages in the LRU queues at which the queue is cleaned	If a value is specified out of the range of 0 through 100 (inclusive), the default of 60.00 percent is set. Fractional values are allowed.
<i>lru_min_dirty</i>	The percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory. Page cleaners might continue cleaning beyond this point under some circumstances.	If a value is specified out of the range of 0 through 100 (inclusive), the default of 80.00 percent is set. Fractional values are allowed.

Usage

Use this function to change the LRU tuning status and also to specify flushing behavior for the current database server session. The ONCONFIG file is not updated.

Use the **AUTO_LRU_TUNING =1** argument to enable automatic LRU tuning for the current session.

Use the **AUTO_LRU_TUNING=0** argument to disable automatic LRU tuning for the current session.

Use the **min** and **max** arguments to reset the **lru_max_dirty** and **lru_min_dirty** values of the BUFFERPOOL configuration parameter. LRU flushing attributes can be set when automatic LRU tuning is either on or off. If the *lru_min_dirty* value is greater to or equal to the *lru_max_dirty* value for a specific buffer pool, the *lru_min_dirty* value is ignored and no error message is returned.

This function is equivalent to the **onmode -wm AUTO_LRU_TUNING** command.

Example

The following example enables automatic LRU tuning and sets LRU flushing attributes:

```
EXECUTE FUNCTION task("onmode","wm","AUTO_LRU_TUNING=1 min=50, max=70");
```

Related reference:

“onmode -wm: Change LRU tuning status” on page 14-27

onmode and Y arguments: Change query plan measurements for a session (SQL administration API)

Use the **onmode** and **Y** arguments with the **admin()** or **task()** function to change the output of query plan measurements for an individual session.

Syntax

```
EXECUTE FUNCTION admin task ("onmode"—,"Y"—,"session_id"—,  
"2"—,"1"—,"0"—);
```

Element	Description	Key Considerations
<i>session_id</i>	The session ID for which to change output to the sqexplain.out file.	

Usage

You can use this function to emulate the SET EXPLAIN statement.

The last argument determines whether or not to record query measurements, including the plan of the query optimizer, an estimate of the number of rows returned, and the relative cost of the query.

Use the **2** argument to enable Informix to send output to the **sqexplain.out.sessionid** file of subsequent query plans, but without executing the query. This setting is equivalent to the SET EXPLAIN ON AVOID_EXECUTE statement.

Use the **1** argument to enable Informix to send output to the **sqexplain.out.sessionid** file. This setting is equivalent to the SET EXPLAIN ON statement for a specific session.

Use the **0** argument to disable the output of query measurements to the **sqexplain.out** file for the current session. This setting is equivalent to the SET EXPLAIN OFF statement.

This function is equivalent to the **onmode -Y** command.

Example

The following example disables the output of query measurements for user session with an ID of 32:

```
EXECUTE FUNCTION task("onmode","Y","32","0");
```

Related reference:

“onmode -Y: Dynamically change SET EXPLAIN” on page 14-28

onmode and z arguments: Terminate a user session (SQL administration API)

Use the **onmode** and **z** arguments with the **admin()** or **task()** function to terminate the specified user session.

Syntax

```
➤➤EXECUTE FUNCTION admin task ("onmode"—,"z"—,"session_ID"—);➤➤
```

Element	Description	Key Considerations
<i>session_ID</i>	The session ID.	The value must be an unsigned integer greater than 0, and must be the session identification number of a currently running session.

Usage

This function is equivalent to the **onmode -z** command.

Example

The following example terminates the user session with an ID of 14:

```
EXECUTE FUNCTION task("onmode","z","14");
```

Related reference:

“onmode -z: Kill a database server session” on page 14-29

onmode and Z arguments: Terminate a distributed transaction (SQL administration API)

Use the **onmode** and **Z** arguments with the **admin()** or **task()** function to terminate the specified distributed transaction. Use this function only if communication between the participating database servers has been lost. If applications are performing distributed transactions, terminating one of the distributed transactions can leave your client/server database system in an inconsistent state.

Syntax

```
➤➤EXECUTE FUNCTION admin task ("onmode"—,"Z"—,"address"—);➤➤
```

Element	Description	Key Considerations
<i>address</i>	The shared-memory address associated with a distributed transaction.	<p>This must be the address of an ongoing distributed transaction that has exceeded the amount of time that the TXTIMEOUT configuration parameter specifies.</p> <p>The <i>address</i> must conform to the operating-system-specific rules for addressing shared-memory. This address is available from onstat -x output.</p>

Usage

This function succeeds only if the distributed transaction has exceeded the amount of time that the TXTIMEOUT configuration parameter specifies.

This function is equivalent to the **onmode -Z** command.

Example

The following example terminates a distributed transaction with an address of 0xa509018:

```
EXECUTE FUNCTION task("onmode","Z","0xa509018");
```

Related reference:

“onmode -Z: Kill a distributed transaction” on page 14-29

onbar argument: Backup the storage spaces (SQL administration API)

Use the **onbar** argument with the **admin()** or **task()** function to backup the storage spaces.

Syntax

```

▶▶ EXECUTE FUNCTION admin task ("onbar backup
whole system logs level number "—) —;
▶▶

```

Element	Purpose	Key considerations
onbar backup	Performs a complete backup of the storage spaces	If you do not specify a level, a level 0 backup is performed.
whole system	Performs a whole-system backup	This is equivalent of issuing the onbar command with the -w option from the command line. If you do not specify a level, a level 0 backup is performed.

Element	Purpose	Key considerations
<code>level number</code>	Specifies the level of backup to perform on storage spaces: <ul style="list-style-type: none"> • 0 for a complete backup. This is the default. • 1 for changes since the last level-0 backup • 2 for changes since the last level-1 backup 	<p>If you request an incremental backup and a level backup has not been performed for a particular storage space, this function backs up that storage space at the previous level.</p> <p>For example, if you request a level-1 backup, and the function finds no level-0 backup, a level-0 backup is made instead.</p> <p>For more information, see Perform an incremental backup.</p> <p>This is equivalent of issuing the onbar command with the -L level option from the command line.</p>
<code>logs</code>	Performs a back of the logical-log files	This is equivalent of issuing the onbar command with the -l option from the command line.

Usage

This function is equivalent to invoking specific options of the **onbar** command to create backups of the storage spaces and logical-log files.

Examples

The following example creates a level 0 backup of the storage spaces:

```
EXECUTE FUNCTION task("onbar backup");
```

The following example creates a level 1 backup of the storage spaces:

```
EXECUTE FUNCTION task("onbar backup level 1");
```

The following example creates a level 1 backup of the logical-log files:

```
EXECUTE FUNCTION task("onbar backup logs");
```


The following example creates a whole system level 0 backup of the storage spaces:

```
EXECUTE FUNCTION task("onbar backup whole system");
```

The following example creates a whole system level 2 backup of the storage spaces:

```
EXECUTE FUNCTION task("onbar backup whole system level 2");
```

Related reference:

 [Back up with ON-Bar \(Backup and Restore Guide\)](#)

 [Backup syntax \(Backup and Restore Guide\)](#)

onsmsync argument: Synchronize with the storage manager catalog (SQL administration API)

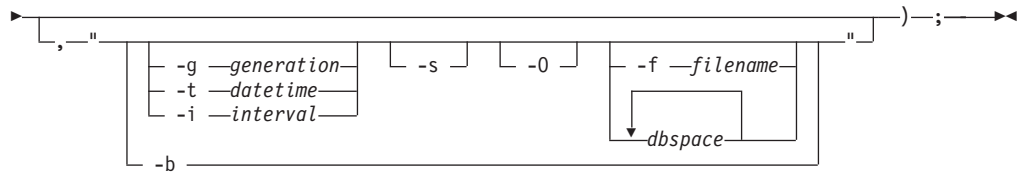
Use the **onsmsync** argument with the **admin()** or **task()** function to synchronize the **sysutils** database and emergency boot file with the storage manager catalog.

Syntax

```

>>> EXECUTE FUNCTION admin task ("onsmsync" )

```



Element	Purpose	Key considerations
no options	Synchronizes the sysutils database and emergency boot file with the storage-manager catalog	None.
-b	Regenerates both the emergency boot file (<code>ixbar.servernum</code>) and the sysutils database from each other.	<p>If the ixbar file is empty or does not exist, onmsmsync -b recreates the ixbar file and populates it from the sysutils tables.</p> <p>If the ixbar is not empty and contains object data, onmsmsync -b updates the sysutils database and the ixbar file so that they are in sync.</p> <p>If the ixbar file has entries and the sysutils database has been rebuilt, but is empty because it does not contain data, the onmsmsync -b option recreates the sysutils data from the ixbar file.</p> <p>The -b element is not used with the other onmsmsync options. Additionally, it does not synchronize with the storage manager.</p>
<i>dbspace</i>	Specifies the storage space or storage spaces to expire	If you enter more than one storage space, use a space to separate the names.
-f <i>filename</i>	Specifies the path name of a file that contains a list of storage spaces to expire	Use this option to avoid entering a long list of storage spaces. The file name can be any valid UNIX or Windows file name.
-g <i>generation</i>	Retains a certain number of versions of each level-0 backup	The latest generation of backups are retained and all earlier ones are expired.
-i <i>interval</i>	Expires all backups older than some period of time	Retains backups younger than this interval. Backups older than interval are not expired if they are needed to restore from other backups after that interval. Use the ANSI or GLS format for the <i>interval</i> : YYYY-MM or DD HH:MM:SS
-s	Skips backups that the storage manager has expired	Use this option to skip synchronizing objects that are already expired from the storage manager. The object expiration will be based on other arguments if the -s option is provided.
-0	Enforces expiration policy strictly	If used with the -t, -g, or -i option, expires all levels of a backup, even if some of them are needed to restore from a backup that occurred after the expiration date. The -0 option does not affect logical-log expiration. See Expire all backups (Backup and Restore Guide).
-t <i>datetime</i>	Expires all backups before a particular date and time	Retains backups younger than this <i>datetime</i> . Backups older than <i>datetime</i> are not expired if they are needed to restore from other backups after that <i>datetime</i> . Use the ANSI or GLS_DATETIME format for <i>datetime</i> .

Usage

This function invokes the **onmsmsync** utility to synchronize the **sysutils** database and emergency boot file with the storage manager catalog.

Example

The following example invokes the **onmsync** utility and specifies that number of backups retained is 1 and all earlier backup versions are expired:

```
EXECUTE FUNCTION task("onmsync", "-g 1");
```

Related reference:

[➡](#) The onmsync utility (Backup and Restore Guide)

ontape archive argument: Backup the data on your database (SQL administration API)

Use the **ontape archive** argument with the **admin()** or **task()** function to create a backup of your database data.

Syntax

```
➡ EXECUTE FUNCTION admin task ("ontape archive" directory file tape level-number "—" "location" "—" "block_size" ) ;
```

Element	Description	Key Considerations
<i>number</i>	Valid levels are: 0, 1, or 2. The default value is 0.	
<i>location</i>	The path to the file or directory or tape device	
<i>block_size</i>	The block size of the device to which ontape writes during a storage-space backup	The default block size is 512 KB.

Usage

This function invokes the ontape utility to create a backup.

There are three devices that you can choose from for the location of the backup:

directory or dir

An existing directory path. The default backup device.

file An existing file.

tape An existing tape device.

Examples


This function creates a level 0 archive in the directory path `/local/informix/backup/`:

```
EXECUTE FUNCTION task("ontape archive", "/local/informix/backup/");
```

This function creates a level 0 archive in the directory path `/local/informix/backup/` with a block size of 256 KB:

```
EXECUTE FUNCTION task("ontape archive directory level 0",  
"/local/informix/backup/", "256");
```

Related reference:

 [Back up with ontape \(Backup and Restore Guide\)](#)

print error argument: Print an error message (SQL administration API)

Use the **print error** argument with the **admin()** or **task()** function to print the message associated with the specified error number.

Syntax

```
►► EXECUTE FUNCTION admin  
task ("print error",—"error_number"—); ►◄
```

Element	Description	Key Considerations
<i>error_number</i>	The error number, without a minus sign.	The <i>error_number</i> must be an existing error number.

Usage

This function is equivalent to the **finderr** utility.

Example

The following example prints the message text for the error number -105:

```
EXECUTE FUNCTION task("print error", "105");  
(expression) ISAM error: bad isam file format.
```

print file info argument: Display directory or file information (SQL administration API)

Use the **print file info** argument with the **admin()** or **task()** function to display information about a directory or a file

Syntax

```
►► EXECUTE FUNCTION admin  
task ("print file info",—"filepath"—); ►◄
```

Element	Purpose	Key considerations
<i>filepath</i>	The path to the directory or file	

Example: File information

The following example shows the argument you would use to print information about the `x` file that is in the `/tmp` directory:

```
execute function task("print file info", "/tmp/x");
```

The following information is returned:

```
(expression) File name      = /tmp/x
              Is File       = 1
              Is Directory   = 0
              Is Raw Device  = 0
              Is Block Device = 0
              Is Pipe        = 0
              File Size      = 554
              Last Access Time = 11/29/2010 21:55:02
              Last Modified Time = 11/29/2010 21:51:45
              Status Change Time = 11/29/2010 21:51:45
              User Id        = 200
              Group id       = 102
              File Flags     = 33206
```

Example: Directory information

The following example shows the argument that you would use to print information about the /tmp directory:

```
execute function task("print file info", "/tmp");
```

The following information is returned:

```
(expression) File name      = /tmp
              Is File       = 0
              Is Directory   = 1
              Is Raw Device  = 0
              Is Block Device = 0
              Is Pipe        = 0
              File Size      = 32768
              Last Access Time = 12/06/2010 11:53:00
              Last Modified Time = 12/06/2010 12:05:53
              Status Change Time = 12/06/2010 12:05:53
              User Id        = 0
              Group id       = 0
              File Flags     = 17407
```

print partition argument: Print partition information (SQL administration API)

Use the **print partition** argument with the **admin()** or **task()** function to print the headers of a specified partition.

Syntax

```
►►—EXECUTE FUNCTION—┐admin┐—————►
                    └task┘

►—(—┐"print partition"——┐,—"partition_number"—)—;—————►◄
   └"print partition full"┘
```

Element	Description	Key Considerations
<i>partition_number</i>	The partition number.	Find the partition numbers in the partnum column of the systables system catalog table.

Usage

Use this function to print a tblspace report for the specified partition.

Run this function with the **full** argument to include index-specific information and page-allocation information by page type for dbspaces.

This function with the **print partition** argument is equivalent to the **oncheck -pt** command.

This function with the **print partition full** argument is equivalent to the **oncheck -pT** command.

Example

The following example prints the headers for a partition with a number of 1048611:
`EXECUTE FUNCTION task("print partition","1048611");`

Related reference:

“oncheck -pt and -pT: Display tblspaces for a Table or Fragment” on page 8-19

rename space argument: Rename a storage space (SQL administration API)

Use the **rename space** argument with the **admin()** or **task()** function to rename a dbspace, blobspace, sbspace, or extspace.

Syntax

```
►►—EXECUTE FUNCTION admin task (—"rename space"—,—"space"—,—"new_name"—►  
►) —; —►
```

Element	Description	Key Considerations
<i>new_name</i>	The new name of the space.	
<i>space</i>	The name of the dbspace, blobspace, sbspace, or extspace that you want to rename.	

Usage

This function is equivalent to the **onspaces -ren** command.

Example

The following example renames a dbspace named **dbsp1** to **dbsp2**:

```
EXECUTE FUNCTION task("rename space","dbsp1","dbsp2");
```

Related reference:

"onspaces -ren: Rename a dbspace, blobspace, sbpace, or extspace" on page 19-24

reset sysadmin argument: Move the sysadmin database (SQL administration API)

Use the **reset sysadmin** argument with the **admin()** or **task()** function to move the **sysadmin** database to the specified dbspace. Moving the **sysadmin** database resets the database back to the original state when it was first created; all data, **command history**, and results tables are lost. Only built-in tasks, sensors, and thresholds remain in the **sysadmin** tables.

Syntax

```
►►EXECUTE FUNCTION admintask("reset sysadmin",,"dbspace");►►
```

Element	Description	Key Considerations
<i>dbspace</i>	The name of the dbspace.	

Usage

This function has no equivalent utility command.

If you specify no *dbspace* as the last argument, this command drops the **sysadmin** database, and then recreates it in the same dbspace where it existed before the function was invoked. All **ph_*** table and **command_history** rows are deleted, and all results tables are dropped.

Examples

The following example moves the **sysadmin** database to a dbspace named **dbsp1**:

```
EXECUTE FUNCTION task("reset sysadmin","dbsp1");
```

The next example destroys the **sysadmin** database, and then recreates it in the same dbspace.

```
EXECUTE FUNCTION admin("reset sysadmin");
```

Except for the built-in tasks, sensors, and thresholds, all data rows are deleted from the **ph_*** tables and all results tables are dropped from **sysadmin** by this function call. The **command_history** table has no rows after the function completes execution.

Related concepts:

Chapter 3, "The sysadmin Database," on page 3-1

restart listen argument: Stop and start a listen thread dynamically (SQL administration API)

Use the **restart listen** argument with the **admin()** or **task()** function to stop and then start an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.

Syntax

```
►► EXECUTE FUNCTION admin (—"restart listen"—,—"server_name"—); ►►  
                                  └─task─┘
```

Element	Description	Key Considerations
<i>server_name</i>	The name of the database server for which you want to stop and restart a listen thread.	

Usage

The definition of the listen thread must exist in the **sqlhosts** file.

If necessary, before you restart a listen thread, revise the **sqlhosts** entry. For example, if a running listen thread is bound to port 7777, you can change the port in the **sqlhosts** file, and then restart the thread.

This function is equivalent to the **onmode -P restart server_name** command.

This function does not update the **sqlhosts** file.

Example

The following command stops and then starts a listen thread for a server named **ids_serv1**:

```
EXECUTE FUNCTION task("restart listen","ids_serv1");
```


Related reference:

“onmode -P: Start, stop, or restart a listen thread dynamically” on page 14-22

“start listen argument: Start a listen thread dynamically (SQL administration API)” on page 21-116

“stop listen argument: Stop a listen thread dynamically (SQL administration API)” on page 21-117

scheduler argument: Stop or start the scheduler (SQL administration API)

Use the **scheduler** argument with the **admin()** or **task()** function to start or stop the scheduler.

Syntax

```
►►EXECUTE FUNCTION admin ("scheduler shutdown")—; task ("scheduler start")—; ►►
```

Usage

Use the **scheduler shutdown** argument to stop the scheduler and deallocate its resources.

Use the **scheduler start** argument to start the scheduler.

This function has no equivalent utility command.

You can view the status of the scheduler threads with the **onstat -g dbc** command.

Example

The following example starts the scheduler after it has been shut down:

```
EXECUTE FUNCTION task("scheduler start");
```

Related reference:

“**onstat -g dbc** command: Print dbScheduler and dbWorker thread statistics” on page 20-73

scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)

Use the **scheduler lmm enable** argument with the **admin()** or **task()** function to start automatic low memory management and to update low memory threshold settings.

Syntax

```
►►EXECUTE FUNCTION admin ("scheduler lmm enable"—, LMM START THRESHOLD—, "start_threshold_size"—, task—) task—; ►►
```

```

└─ LMM STOP THRESHOLD—,—"stop_threshold_size"—,—
└─ LMM IDLE TIME—,—"minimum_amount_of_time"—,—
) —;

```

Element	Description	Key Considerations
<i>start_threshold_size</i>	The amount of free memory that you want the database server to maintain. If the amount of memory falls below the <i>start_threshold_size</i> , the server automatically frees memory and terminates applications.	<p>The value can be expressed as either a percentage of the value of the SHMTOTAL configuration parameter or as a specific amount. If the value is less than 50, it is considered a percentage. The resulting value of the input parameter must be more than 5 MB and less than 95 MB.</p> <p>The default value is 5 MB.</p> <p>There must be a minimum 5 MB difference between the LMM START THRESHOLD and LMM STOP THRESHOLD values</p>
<i>stop_threshold_size</i>	The amount of free memory that you want the database server to have, before the server stops automatically freeing memory and terminating applications.	<p>The value can be expressed as either a percentage of the value of the SHMTOTAL configuration parameter or as a specific amount. If the value is less than 50, it is considered a percentage. The resulting value of the input parameter must be more than 10 MB and less than 100 MB. The value must also be at least 5 MB more than the LMM START THRESHOLD.</p> <p>The default value is 10 MB.</p>
<i>minimum_amount_of_time</i>	The amount of time in seconds that defines a session as idle	<p>The value must be between 1 and 86400.</p> <p>The default value is 300 seconds.</p>

Usage

If you enable automatic low memory management and configure the database server to use a percentage of the value specified in the SHMTOTAL configuration parameter for the start and stop thresholds, use caution when changing the value of the SHMTOTAL configuration parameter. Changing the value of the SHMTOTAL configuration parameter value can cause the configuration of automatic low memory management to become invalid, forcing Informix to use the default settings.

You use the **scheduler lmm disable** argument with the **admin()** or **task()** function to stop the current and subsequent low memory management processes.

The database server stores automatic low memory management settings in the **ph_threshold** table.

You can view low memory management settings and recent activity with the **onstat -g lmm** command.

When low memory management is triggered, the database server performs these tasks, in order:

1. The database server terminates sessions starting one at a time from the session with largest amount of idle time and continuing as necessary to the session with smallest amount of idle time that exceeds the amount specified in the LMM IDLE TIME setting. The server stops terminating sessions when the LMM STOP THRESHOLD is reached.
2. The database server terminates sessions starting with the session using the most memory and continuing as necessary to the session using the smallest amount of memory until the LMM STOP THRESHOLD is reached.
3. The database server performs memory reconfiguration by setting the VP_MEMORY_CACHE configuration parameter to 0 and running the **onmode -F** command to free unused shared memory segments.

When the low memory management operations are complete, the low memory manager returns to monitoring mode and restores the memory configuration of the database server by setting the VP_MEMORY_CACHE configuration parameter back to its original value.

Example

The following example specifies that when the database server has 10 MB or less of free memory, the server will start automatic low memory management to stop applications and to free memory. The example also specifies that a session is considered idle if it has not run for 300 seconds, and the example specifies that the server will stop automatic low memory management when the server has 20 MB or more of free memory.

```
EXECUTE FUNCTION task("scheduler lmm enable",
    "LMM START THRESHOLD", "10MB",
    "LMM STOP THRESHOLD", "20MB",
    "LMM IDLE TIME", "300");
```

Example of the SHMTOTAL configuration parameter impacting low memory management threshold settings:

Suppose you set the SHMTOTAL configuration parameter to 1000000 (1000 MB or 1 GB), the LMM START THRESHOLD to 2, and the LMM STOP THRESHOLD to 3. Because any value that is less than 50 is a percentage of the value of SHMTOTAL, the actual LMM START THRESHOLD is 20000 (20 MB) and the actual LMM STOP THRESHOLD is 30000 (30 MB).

The database server begins managing low memory when the remaining free memory is 20 MB or less and stop managing memory when the amount of free memory is 30 MB or greater.

Suppose you decide to change the value of the SHMTOTAL configuration parameter because you know now that you don't need as much memory and you

want memory to be available to the operating system. You set the value of SHMTOTAL to 250000 (250 MB). This changes the actual LMM START THRESHOLD to 5000 (5 MB) and the LMM STOP THRESHOLD to 7500 (7.5 MB). The LMM STOP THRESHOLD is now invalid because there must be a minimum 5 MB difference between the LMM START THRESHOLD and LMM STOP THRESHOLD values. The LMM STOP THRESHOLD value must also be at least 10 MB.

You might have decided that a 10 MB difference is the right amount for your system. But at 5 MB, the database server could spend too much time spent on low memory management processes and this could cause performance problems.

Related concepts:

 Configure the server response when memory is critically low (Administrator's Guide)

Related reference:

“scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)”

“**onstat -g lmm** command: Print low memory management information” on page 20-111

“LOW_MEMORY_MGR configuration parameter” on page 1-89

“SHMTOTAL configuration parameter” on page 1-130

“VP_MEMORY_CACHE_KB Configuration Parameter” on page 1-156

scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)

Use the **scheduler lmm disable** argument with the **admin()** or **task()** function to stop the current and subsequent invocations of automatic low memory management.

Syntax

```

▶▶—EXECUTE FUNCTION—

|       |
|-------|
| admin |
| task  |

—(—"scheduler lmm disable"—,—)—;—▶▶

```

Usage

If automatic low memory management is enabled, you can disable it by specifying:
EXECUTE FUNCTION task("scheduler lmm disable");

You use the **scheduler lmm enable** argument with the **admin()** or **task()** function to start automatic low memory management and update threshold settings.

You can view information about automatic low memory management settings and recent activity with the **onstat -g lmm** command.

Related reference:

“scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)” on page 21-101

“**onstat -g lmm** command: Print low memory management information” on page 20-111

“LOW_MEMORY_MGR configuration parameter” on page 1-89

set chunk argument: Change the status of a chunk (SQL administration API)

Use the **set chunk** argument with the **admin()** or **task()** function to change the status of a blobspace, dbspace, or sbspace to online or offline.

Syntax

```
►►—EXECUTE FUNCTION—admintask—("set chunk offline""set chunk online")—►  
►—,"space name"—,"path"—,"offset"—)—;—►
```

Element	Description	Key Considerations
<i>space name</i>	The name of the blobspace, dbspace, or sbspace.	
<i>path</i>	The disk partition or unbuffered device of the chunk.	
<i>offset</i>	The offset, in kilobytes, into the disk partition or unbuffered device to reach the chunk.	See “admin() and task() Argument Size Specifications” on page 21-2.

Usage

The chunk must be in a mirrored pair, or a non-primary chunk within a noncritical dbspace.

Use the **set chunk offline** argument to change the status of the chunk to offline.

Use the **set chunk online** argument to change the status of the chunk to online.

This function is equivalent to the **onspaces -s** command.

Example

The following example changes the status of a chunk to online:

```
EXECUTE FUNCTION task("set chunk online","dbs1","/dev/raw_dev1","0");  
Database selected.
```

```
(expression) Chunk status successfully changed.  
Chunk number 2 "/dev/raw_dev1" -- Online
```

```
1 row(s) retrieved.
```


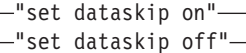
Related reference:

“onspaces -s: Change status of a mirrored chunk” on page 19-26

set dataskip argument: Start or stop skipping a dbspace (SQL administration API)

Use the **set dataskip** argument with the **admin()** or **task()** function to specify whether the database server skips a dbspace that is unavailable during the processing of a transaction.

Syntax

```
►►EXECUTE FUNCTION  ("set dataskip on" ,—"dbspace"—);►►
```

Element	Description	Key Considerations
<i>dbspace</i>	The name of the dbspace to begin or to stop skipping.	

Usage

Run this function to update the value of the DATASKIP configuration parameter, which specifies whether the database server skips a dbspace that is unavailable (for example, due to a media failure) in the course of processing a transaction.

Use the **set dataskip on** argument to begin skipping the specified dbspace when it is down.

Use the **set dataskip off** argument to stop skipping the specified dbspace.

This function is equivalent to the **onspaces -f** command.

Example

The following example skips the dbspace named **dbsp1** if it is down:

```
EXECUTE FUNCTION task("set dataskip on","dbsp1");
```

Related reference:

“onspaces -f: Specify DATASKIP parameter” on page 19-21

set index compression argument: Change index page compression (SQL administration API)

Use the **set index compression** argument with the **admin()** or **task()** function to modify the level at which two partially used index pages are merged.

Syntax

```
►►EXECUTE FUNCTION  (—————►
```

```
►—"set index compression"—,"partition_number"—,["med"
                                                    "default"
                                                    "high"
                                                    "low"]—);—►
```

Element	Description	Key Considerations
<i>partition_number</i>	The partition number.	Find the partition numbers in the partnum column of the systables system catalog table.

Usage

Use this function to adjust index page compression. The pages are merged if the data on those pages totals a set level. To optimize space and transaction processing, you can lower the compression level if your indexes grow quickly. You can increase the level if your indexes have few delete and insert operations or if batch updates are performed.

Use the **low** argument if you expect an index to grow quickly with frequent splits.

Use the **med** or **default** argument if an index has moderate growth or changes.

Use the **high** argument if an index is 90 percent or more read-only or does not have many changes.

This function is equivalent to the **onmode -C** command and the **compression** option of the BTSCANNER configuration parameter.

Example

The following example sets index compression for a partition to high:

```
EXECUTE FUNCTION task("set index compression","1048611","high");
```

Related reference:

"onmode -C: Control the B-tree scanner" on page 14-5

set onconfig memory argument: Temporarily change a configuration parameter (SQL administration API)

Use the **set onconfig memory** argument with the **admin()** or **task()** function to dynamically update the value of a configuration parameter in memory.

Syntax

```
►►—EXECUTE FUNCTION—[admin—
                       task—]—(—————►
►—"set onconfig memory"—,"configuration_parameter"—,"value"—);—►
```

Element	Description	Key Considerations
<i>configuration_parameter</i>	The name of a configuration parameter.	<p>The configuration parameter must be one that you can update dynamically.</p> <p>The list of configuration parameters that you can update dynamically is the same as for the onmode -wf command.</p>
<i>value</i>	The new value or values for the configuration parameter.	<p>The new value or values must be valid for the configuration parameter.</p> <p>The format of the new value must conform exactly to the syntax for that configuration parameter.</p>

Usage

Use this function to temporarily update the value of a configuration parameter that can be dynamically updated. The new value takes affect immediately. The new value is not written to the **onconfig** file and is lost when the database server is restarted.

This function is equivalent to the **onmode -wm** command.

Example

The following example sets the value of the DYNAMIC_LOGS configuration parameter to 2 for the current session:

```
EXECUTE FUNCTION task("set onconfig memory","DYNAMIC_LOGS","2");
```

Related reference:

"onmode -wf, -wm: Dynamically change certain configuration parameters" on page 14-25

set onconfig permanent argument: Permanently change a configuration parameter (SQL administration API)

Use the **set onconfig permanent** argument with the **admin()** or **task()** function to dynamically update the value of a configuration parameter in the **onconfig** file.

Syntax

```

▶▶ EXECUTE FUNCTION admin (—————▶
task
▶ "set onconfig permanent"—,"configuration_parameter"—,"value"—)——▶▶

```


Element	Description	Key Considerations
<i>configuration_parameter</i>	The name of a configuration parameter.	<p>The configuration parameter must be one that you can update dynamically.</p> <p>The list of configuration parameters that you can update dynamically is the same as for the onmode -wf command.</p>
<i>value</i>	The new value or values for the configuration parameter.	<p>The new value or values must be valid for the configuration parameter.</p> <p>The format of the new value must conform exactly to the syntax for that configuration parameter.</p>

Usage

Use this function to permanently update the value of a configuration parameter. The new value takes affect immediately and persists in the **onconfig** file after the server restarts.

This function is equivalent to the **onmode -wf** command.

Example

The following example sets the value of the DYNAMIC_LOGS configuration parameter to 2 in the **onconfig** file:

```
EXECUTE FUNCTION task("set onconfig permanent","DYNAMIC_LOGS","2");
```

Related reference:

“onmode -wf, -wm: Dynamically change certain configuration parameters” on page 14-25

set sbspace accesstime argument: Control access time tracking (SQL administration API)

Use the **set sbspace accesstime** argument with the **admin()** or **task()** function to start or stop tracking the time of access for all smart large objects stored in the sbspace.

Syntax

```

▶▶ EXECUTE FUNCTION admin (task ("set sbspace accesstime off"
"set sbspace accesstime on"
▶▶ ,—"sbspace"—) ) ;

```

Element	Description	Key Considerations
<i>sbspace</i>	The name of the sbspace.	

Usage

Use the **set sbospace accesstime off** argument to turn off tracking of access times.

Use the **set sbospace accesstime on** argument to turn on tracking of access times for all smart large objects stored in the sbospace.

This function is equivalent to the **onspaces -ch** command.

Example

The following example turns off tracking of access times for the sbospace named **sbsp1**:

```
EXECUTE FUNCTION task("set sbospace accesstime off","sbsp1");
```

Related reference:

“onspaces -ch: Change sbospace default specifications” on page 19-17

“create sbospace with accesstime argument: Create an sbospace that tracks access time (SQL administration API)” on page 21-36

set sbospace avg_lo_size argument: Set the average size of smart large objects (SQL administration API)

Use the **set sbospace avg_lo_size** argument with the **admin()** or **task()** function to specify an expected average size of the smart large objects in the specified sbospace so that the database server can calculate the size of the metadata area.

Syntax

```
►► EXECUTE FUNCTION admin  
task (—————►  
  
► "set sbospace avg_lo_size"—,—"sospace"—,—"size"—) —;————►◄
```

Element	Description	Key Considerations
<i>sospace</i>	The name of the sbospace.	
<i>size</i>	The average size, in kilobytes, of the smart large object stored in the sbospace.	Windows: 4 to 2**31 UNIX: 2 to 2**31

Usage

This function is equivalent to the **onspaces -ch** command.

Example

The following example sets the expected average size of smart large objects in the sbospace named **sbsp1** to 8 KB:

```
EXECUTE FUNCTION task("set sbospace avg_lo_size","sbsp1","8");
```

"onspaces -ch: Change sbospace default specifications" on page 19-17

Use the **set sbospace logging** argument with the **admin()** or **task()** function to specify whether the database server logs changes to the user data area of the sbospace.

```

▶▶EXECUTE FUNCTION      admin      (—"set sbospace logging on"
                                └task└—"set sbospace logging off"
▶▶,—"sbospace"—);—

```

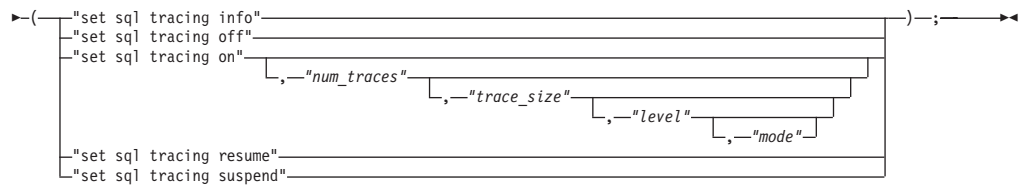
“onspaces -ch: Change sbospace default specifications” on page 19-17

Use the **set sql tracing** argument with the **admin()** or **task()** function to set global SQL tracing.

```

    >>EXECUTE FUNCTION admin

```



Element	Description	Key Considerations
<i>level</i>	The tracing level. The default is low .	Possible values are: <ul style="list-style-type: none"> • low • med • high
<i>mode</i>	Whether all or selected users are traced.	Possible modes are: <ul style="list-style-type: none"> • global • user
<i>num_traces</i>	The number of SQL statements to trace. The default value is 1000.	
<i>trace_size</i>	The number of kilobytes for the size of the trace buffer. If this buffer size is exceeded, the database server discards saved data. The default size is 2 KB.	

Usage

Use this function to reset the value of the SQLTRACE configuration parameter.

Use the **set sql tracing info** argument to display the state of global SQL tracing.

Use the **set sql tracing off** argument to turn off global SQL tracing.

Use the **set sql tracing on** argument to turn on global SQL tracing. Optionally specify the tracing level and mode:

- Use the **low** argument to capture statement statistics, statement text, and statement iterators.
- Use the **med** argument to capture all of the information included in low-level tracing, plus table names, the database name, and stored procedure stacks.
- Use the **high** argument to capture all of the information included in medium-level tracing, plus host variables.
- Use the **global** argument to enable tracing for all users.
- Use the **user** argument to enable tracing for those users who have tracing enabled by the **set sql tracing user** argument.

Use the **set sql tracing resume** argument to restart SQL tracing when it has been suspended.

Use the **set sql tracing suspend** argument to pause SQL tracing without deallocating any resources.

Example

The following example starts a high level of global tracing for 1500 SQL statements into a 4 KB trace buffer:

```
EXECUTE FUNCTION task("set sql tracing on","1500","4","high","global");
```

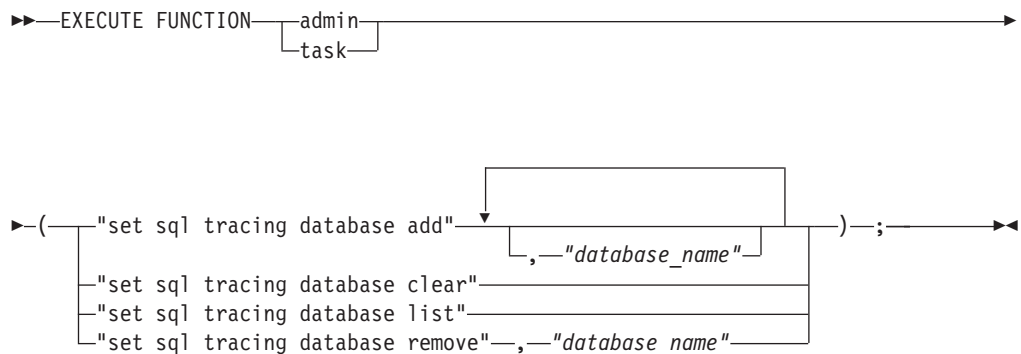
The following example pauses SQL tracing:

```
EXECUTE FUNCTION task("set sql tracing suspend");
```

set sql tracing database argument: Change database tracing (SQL administration API)

Use the **set sql tracing database** argument with the **admin()** or **task()** function to start or stop tracing for a database, or list which databases are being traced.

Syntax



Element	Description	Key Considerations
<i>database_name</i>	The name of the database.	Specify one database name.

Usage

Use the **set sql tracing database add** argument to specify tracing for one or more databases, rather than for all databases. The default is all databases. Specify up to six arguments in a single **admin()** or **task()** function. The maximum number of database names that can be set is 16.

Use the **set sql tracing database clear** argument to clear all databases from the list of databases being traced. Returns tracing back to the default of all databases.

Use the **set sql tracing database list** argument to list the databases that are being traced.

Use the **set sql tracing database remove** argument to remove a single database from the list of databases being traced.

When you use the **set sql tracing database** argument, you can specify only the name of one database. While you can have a maximum of 16 database names, you must specify each additional database name in separate function calls. Each time you call the function, the function adds another database to the list, until the list contains 16 databases.

Example

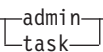
The following example sets SQL tracing for three databases with the names **db1**, **db2** and **db3**:

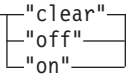
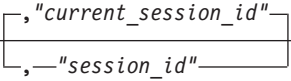
```
EXECUTE FUNCTION task("set sql tracing database add","db1");
EXECUTE FUNCTION task("set sql tracing database add","db2");
EXECUTE FUNCTION task("set sql tracing database add","db3");
```

set sql tracing session argument: Control tracing for a session (SQL administration API)

Use the **set sql tracing session** argument with the **admin()** or **task()** function to change SQL tracing for the current session.

Syntax

```
►►—EXECUTE FUNCTION——————►

►—("set sql tracing session"—,,—————►

►—);—————►◄
```

Element	Description	Key Considerations
<i>current_session_id</i>	The ID of the current session. This is the default session ID.	
<i>session_id</i>	The ID of the session to which this command applies.	

Usage

Use the **clear** argument to clear any global tracing overrides. The session will conform to the global tracing policy.

Use the **off** argument to turn off tracing for the session, even if the global tracing policy is set to enable tracing.

Use the **on** argument to turn on tracing for the session, even if the global tracing policy is set to disable tracing.

Example

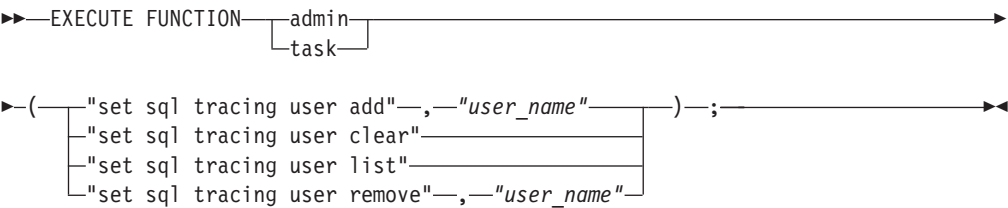
The following example stops tracing for the current session:

```
EXECUTE FUNCTION task("set sql tracing session","off");
```

set sql tracing user argument: Control tracing for users (SQL administration API)

Use the **set sql tracing user** argument with the **admin()** or **task()** function to change SQL tracing for users.

Syntax



Element	Description	Key Considerations
<i>user_name</i>	The name of the user.	

Usage

Use the **set sql tracing user add** argument to specify tracing for a specific user.

Use the **set sql tracing user clear** argument to remove all users from the tracing list.

Use the **set sql tracing user list** argument to list the users that are being traced.

Use the **set sql tracing user remove** argument to remove a single user from the list of users being traced.

Example

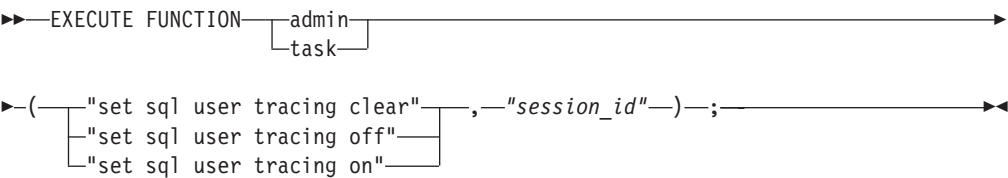
The following example stops tracing SQL statements for the user named **fred**:

```
EXECUTE FUNCTION task("set sql tracing user remove","fred");
```

set sql user tracing argument: Set global SQL tracing for a user session (SQL administration API)

Use the **set sql user tracing** argument with the **admin()** or **task()** function to set the mode of global SQL tracing for a specified user session.

Syntax



Element	Description	Key Considerations
<i>session_id</i>	The ID for the session.	

Usage

Use the **set sql user tracing clear** to clear user tracing flags for the specified user session so that it adheres to the global tracing policy.

Use the **set sql user tracing off** to disable SQL tracing for a user session even if the global mode is ON.

Use the **set sql user tracing on** to enable user SQL tracing for a user session. Even if the global tracing mode is OFF, SQL statements for this user session are traced.

Example

The following example starts tracing for the session with the ID of 18:

```
EXECUTE FUNCTION task("set sql user tracing on","18");
```

start listen argument: Start a listen thread dynamically (SQL administration API)

Use the **start listen** argument with the **admin()** or **task()** function to start an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.

Syntax

```

▶▶EXECUTE FUNCTION admin task ("start listen"—,"server_name"—);

```

Element	Description	Key Considerations
<i>server_name</i>	The name of the database server for which you want to start a listen thread.	

Usage

The definition of the listen thread must exist in the **sqlhosts** file for the server. If the definition of the listen thread does not exist in the **sqlhosts** file, you must add it before you can start the listen thread dynamically. For information on adding listen threads, see the *IBM Informix Administrator's Guide*.

This function does not update the **sqlhosts** file.

This function is equivalent to the **onmode -P start server_name** command.

Example

The following command starts a new listen thread for a server named **ids_serv2**:

```
EXECUTE FUNCTION task("start listen","ids_serv2");
```


Related reference:

“onmode -P: Start, stop, or restart a listen thread dynamically” on page 14-22

“stop listen argument: Stop a listen thread dynamically (SQL administration API)”

“restart listen argument: Stop and start a listen thread dynamically (SQL administration API)” on page 21-100

start mirroring argument: Starts storage space mirroring (SQL administration API)

Use the **start mirroring** argument with the **admin()** or **task()** function to start mirroring for a specified dbspace, blobspace, or sbspace.

Syntax

```
►►EXECUTE FUNCTION admin  
task ("start mirroring",—"space"—);►►
```

Element	Description	Key Considerations
<i>space</i>	The name of the blobspace, dbspace, or sbspace.	

Usage

This function is equivalent to the **onspaces -m** command.

Example

The following example starts mirroring for the dbspace named **dbbsp1**:

```
EXECUTE FUNCTION task("start mirroring","dbbsp1");
```

Related reference:

“onspaces -m: Start mirroring” on page 19-22

stop listen argument: Stop a listen thread dynamically (SQL administration API)

Use the **stop listen** argument with the **admin()** or **task()** function to stop an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.

Syntax

```
►►EXECUTE FUNCTION admin  
task ("stop listen",—"server_name"—);►►
```

Element	Description	Key Considerations
<i>server_name</i>	The name of the database server for which you want to stop a listen thread.	

Usage

The definition of the listen thread must exist in the **sqlhosts** file.

This function does not update the **sqlhosts** file.

This function is equivalent to the **onmode -P stop server_name** command.

Example

The following command stops a listen thread for a server named **ids_serv3**:

```
EXECUTE FUNCTION task("stop listen","ids_serv3");
```

Related reference:

“onmode -P: Start, stop, or restart a listen thread dynamically” on page 14-22

“start listen argument: Start a listen thread dynamically (SQL administration API)” on page 21-116

“restart listen argument: Stop and start a listen thread dynamically (SQL administration API)” on page 21-100

stop mirroring argument: Stops storage space mirroring (SQL administration API)

Use the **stop mirroring** argument with the **admin()** or **task()** function to stop mirroring for a specified dbspace, blobspace, or sbspace.

Syntax

►►EXECUTE FUNCTION admin
task(—"stop mirroring"—,"space")—;—►►

Element	Description	Key Considerations
<i>space</i>	The name of the blobspace, dbspace, or sbspace.	

Usage

This function is equivalent to the **onspaces -r** command.

Example

The following example stops mirroring for the dbspace with the name of **dbsp1**:

```
EXECUTE FUNCTION task("stop mirroring","dbsp1");
```

Related reference:

“onspaces -r: Stop mirroring” on page 19-24

storagepool add argument: Add a storage pool entry (SQL administration API)

Use the **storagepool add** argument with the **admin()** or **task()** function to add an entry to the *storage pool* (a collection of available raw devices, cooked files, or directories that Informix can use to automatically add space to an existing storage space).

Syntax

```
►►EXECUTE FUNCTION admin task (—"storagepool add"—,"path"—————►  
►,—"begin_offset"—,—"total_size"—,—"chunk_size"—,—"priority"—)——►►
```

Element	Description	Key Considerations
<i>path</i>	The path for the file, directory, or device that the server can use when additional storage space is required.	You do not need to add a final slash ("/") to a directory name. You can use an environment variable in the path if the variable is in your environment when the oninit command runs.
<i>begin_offset</i>	The offset in kilobytes into the device where Informix can begin allocating space.	If you specified a path to a directory, you must specify 0 as the offset.
<i>total_size</i>	The total space available to Informix in this entry. The server can allocate multiple chunks from this amount of space.	Be sure to specify 0 for the total size of a directory. If you specify a value that is not zero for a directory, the SQL administration API command returns an error. If you specify 0 for a file or device, the server will allocate one extendable chunk from the entry.
<i>chunk_size</i>	The minimum size of a chunk that can be allocated from the device, file, or directory.	The smallest chunk that you can create is 1000 K. Therefore, the minimum chunk size that you can specify is 1000 K. See “admin() and task() Argument Size Specifications” on page 21-2.
<i>priority</i>	The priority of the directory, file, or device when the server searches through the storage pool for space. 1 = High priority 2 = Medium priority 3 = Low priority	The server tries to allocate space from a high-priority entry before it allocates space from a lower priority entry.

Usage

The server uses entries in the storage pool if necessary to add a new chunk to a storage space.

When you add an entry to the storage pool, you might want some control over how that entry is used. For example, to reduce the number of chunks in an instance, you might want only large chunks of space to be allocated from a particular raw device, and might not want the chunks to be extendable. In this case, configure the chunk size for that storage pool entry to be large.

You can add the following types of entries to the storage pool:

- A fixed-length raw device
- A fixed-length cooked file
- An extendable raw device (for extending the size of a chunk)
- An extendable cooked file (for extending the size of a chunk)
- A directory

A storage pool entry that is a directory is always categorized as extendable, because it does not have a total size. If new chunks are automatically created in the directory, the server marks those chunks as extendable. When you add a storage pool entry that is a directory, you might want a small chunk size, because the server can extend any chunk created in the directory and smaller chunk sizes can reduce the amount of wasted space in the instance.

If a storage pool entry is on a High-Availability Data Replication (HDR) primary server, the same path in the entry must be available on all secondary servers in the HDR cluster.

The default units for sizes and offsets are kilobytes. However, you can specify information in any of the ways shown in the following examples:

- "100000"
- "100000 K"
- "100 MB"
- "100 GB"
- "100 TB"

Examples

Adding a storage pool entry for a directory

The following command adds a directory named `/region2/dbspaces` with a beginning offset of 0, a total size of 0, an initial chunk size of 20 megabytes, and a high priority:

```
DATABASE sysadmin;  
EXECUTE FUNCTION task("storagepool add", "/region2/dbspaces", "0", "0",  
"20000", "1");
```

Adding a storage pool entry for a fixed-length raw device

The following command adds a fixed-length raw device with the path name `/dev/raw/device1` and a total of 500 megabytes of space to the storage pool. The

command specifies a beginning offset of 50 megabytes, a total size of 10 gigabytes, a minimum of 100 megabytes to allocate to a chunk, and a low priority.

```
EXECUTE FUNCTION task("storagepool add", "/dev/rawdevice1", "50 MB",  
"10 GB", "100 MB", "3");
```

Adding a storage pool entry for a fixed-length cooked file

The following command adds a fixed-length cooked file and 1 gigabyte of space to the storage pool. The command specifies a beginning offset of 0, a total size of 1000000 kilobytes, a minimum of 50000 kilobytes to allocate to a chunk, and a medium priority:

```
EXECUTE FUNCTION task("storagepool add", "/ifmx_filesystem/storage/cooked7",  
"0", "1000000", "50000", "2");
```

When adding this entry, the server tries to increase the size of the cooked7 file to 1 gigabyte. If the server cannot increase the size because the file system is full, the server returns an error message and does not add the entry to the storage pool.

Informix uses part of the cooked file initially, but can use more of the device as necessary as spaces fill.

Adding a storage pool entry for an extendable cooked file

The following command adds a cooked file with the path name /ifmx/CHUNKFILES/cooked2. If the server uses this entry, the server creates one chunk with an initial size of 1 GB, and the server automatically marks the chunk as extendable.

```
EXECUTE FUNCTION task("storagepool add", "/ifmx/CHUNKFILES/cooked2",  
"0", "0", "1 GB", "2");
```

Adding a storage pool entry with an environment variable in the path

The following example includes an environment variable in the path. The variable was present in the server environment when the **oninit** command ran.

```
EXECUTE FUNCTION task("storagepool add", "$DBSDIR/chunk1",  
"0", "100000", "20000", "2");
```

Related reference:

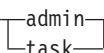
“storagepool modify argument: Modify a storage pool entry (SQL administration API)” on page 21-122

“storagepool delete argument: Delete one storage pool entry (SQL administration API)”

“storagepool purge argument: Delete storage pool entries (SQL administration API)” on page 21-124

storagepool delete argument: Delete one storage pool entry (SQL administration API)

Use the **storagepool delete** argument with the **admin()** or **task()** function to delete an entry from the storage pool.

```
►► EXECUTE FUNCTION  ("—storagepool delete—", —————>
```

```
►"—entry_id—"—);—————►
```

Element	Description	Key Considerations
<i>entry_id</i>	The ID of the storage pool entry.	The storagepool table in the sysadmin database contains a column that shows the ID of each entry in the storage pool.

Usage

Delete a storage pool entry if you do not want the server to continue to use the entry when expanding a storage space.

To delete all storage pool entries, storage pool entries that have a status of Full, or storage pool entries that have a status of Error, use the SQL administration API **storagepool purge** command. (The **storagepool** table in the **sysadmin** database contains a column that shows the status of each entry in the storage pool.)

Example

The following command deletes the storage pool entry with an entry id of 13:

```
EXECUTE FUNCTION task("storagepool delete", "13");
```

Related reference:

"storagepool add argument: Add a storage pool entry (SQL administration API)" on page 21-119

"storagepool modify argument: Modify a storage pool entry (SQL administration API)"

"storagepool purge argument: Delete storage pool entries (SQL administration API)" on page 21-124

storagepool modify argument: Modify a storage pool entry (SQL administration API)

Use the **storagepool modify** argument with the **admin()** or **task()** function to modify an entry for a directory, cooked file, or raw device that Informix can use when additional storage space is required.

Syntax

```
►►EXECUTE FUNCTION—

|       |
|-------|
| admin |
| task  |

—(—————►
```

```
►"—"storagepool modify"—,—"—entry_id"—,——————►
```

```
►—"new_total_size"—,——"new_chunk_size"—,——"new_priority"—)—;—————►
```

Element	Description	Key Considerations
<i>entry_id</i>	The ID of the storage pool entry.	The storagepool table in the sysadmin database contains a column that shows the ID of each entry in the storage pool.

Element	Description	Key Considerations
<i>new_total_size</i>	The new amount of total space available to Informix in this entry. The server can allocate multiple chunks from this amount of space.	Be sure to specify 0 for the total size of a directory. If you specify a value that is not zero for a directory, the SQL administration API command returns an error. If you specify 0 for a file or device, the server allocates one extendable chunk from the entry.
<i>new_chunk_size</i>	The minimum size of a chunk that can be allocated from the device, file, or directory.	The smallest chunk that you can create is 1000 K. Therefore, the minimum chunk size that you can specify is 1000 K. See “admin() and task() Argument Size Specifications” on page 21-2.
<i>new_priority</i>	The priority of the directory, file, or device when the server searches through the storage pool for space. 1 = High priority 2 = Medium priority 3 = Low priority	The server attempts to allocate space from a high-priority entry before it allocates space from a lower priority entry.

Usage

Sometimes you might want to change a storage pool entry. For example, you might want to increase the total size of the storage pool when it runs out of space, or you might want to change the chunk size or the priority. When you change the entry, include the total size, chunk size, and priority even if you do not want to change all of these values.

You cannot modify the path or the beginning offset of a storage pool entry. If you want to change either of those values, you must delete the storage pool entry and add an entry with the new path or beginning offset.

If a storage pool entry is on a High-Availability Data Replication (HDR) primary server, the same path in the entry must be available on all secondary servers in the HDR cluster.

The default units for storage pool sizes and offsets are kilobytes. However, you can specify information in any of the ways shown in the following examples:

- "100000"
- "100000 K"
- "100 MB"
- "100 GB"
- "100 TB"

Examples

The following command changes the total size, chunk size, and priority of the storage pool entry that has an ID of 4 to 10 gigabytes, 10 megabytes, and a medium priority.

```
EXECUTE FUNCTION task("storagepool modify", "4", "10 GB", "10000", "2");
```

Suppose that you add an entry to the storage pool and the entry has a path of (/dev/IDS/chunk2), an offset of 0, a total size of 100 megabytes, a minimum chunk size of 100 megabytes, and a priority of 2. Before Informix allocates any space from this entry, you use **onspaces** to manually add a 50 megabyte chunk with the same path (/dev/IDS/chunk2), and an offset of 50 megabytes. The server only detects the overlap when it attempts to use this entry to automatically create a chunk. At that time, the server marks the entry with an "Error" status and attempts to use another entry to create the chunk.

You can correct the problem by changing the total size of the storage pool entry (for example, for entry 2), to 50 megabytes and by changing the minimum chunk size of the entry to 50 megabytes, as follows:

```
EXECUTE FUNCTION task("storagepool modify", "2", "50 MB", "50 MB", "2");
```

Related reference:


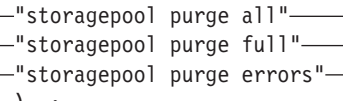
"storagepool add argument: Add a storage pool entry (SQL administration API)" on page 21-119

"storagepool delete argument: Delete one storage pool entry (SQL administration API)" on page 21-121

"storagepool purge argument: Delete storage pool entries (SQL administration API)"

storagepool purge argument: Delete storage pool entries (SQL administration API)

Use the **storagepool purge** argument with the **admin()** or **task()** function to delete all storage pool entries, storage pool entries that have a status of Full, or storage pool entries that have a status of Error.

```
►► EXECUTE FUNCTION  (  ) ; ►
```

Usage

Use the **storagepool purge all** argument to delete all entries in the storage pool.

Use the **storagepool purge full** argument to delete all storage pool entries that have a status of Full.

Use the **storagepool purge errors** argument to delete all storage pool entries that have a status of Error.

The **storagepool** table in the **sysadmin** database contains a column that shows the status of each entry in the storage pool.

Example

The following command deletes all storage pool entries that have a status of Full:
`EXECUTE FUNCTION task("storagepool purge full");`

Related reference:

“storagepool add argument: Add a storage pool entry (SQL administration API)” on page 21-119

“storagepool modify argument: Modify a storage pool entry (SQL administration API)” on page 21-122

“storagepool delete argument: Delete one storage pool entry (SQL administration API)” on page 21-121

Compress and Uncompress Operations (SQL administration API)

You can compress and uncompress the data in a table or in table fragments with SQL administration API **admin()** or **task()** functions and arguments. Compression operations apply only to the contents of data rows and the images of those data rows that appear in logical log records.

The built-in SQL administration API **admin()** or **task()** functions are defined in the **sysadmin** database of each Informix instance. By default, only user **informix** can invoke these functions. If Connect privilege on the **sysadmin** database is granted to user **root** or to **DBSA** group members, they too can invoke the SQL administration API **admin()** or **task()** functions when they are connected directly or remotely to the **sysadmin** database.

The SQL administration API **admin()** or **task()** command arguments that you can use for compress and uncompress operations in tables and table fragments are:

table *compression parameters*

Performs various compression operations to all fragments of a specified table. For more information, see “table or fragment arguments: Compress data and optimize storage (SQL administration API)” on page 21-126.

fragment *compression parameters*

Performs various compression operations to a single fragment or a specified set of fragments that belong to a specific table. For more information, see “table or fragment arguments: Compress data and optimize storage (SQL administration API)” on page 21-126.

compression *purge_dictionary*

Deletes all inactive compression dictionaries or all inactive compression dictionaries that were created before a date that you specify. For more information, see “purge compression dictionary arguments: Remove compression dictionaries (SQL administration API)” on page 21-132.

Table and fragment compression operations include creating compression dictionaries, estimating compression ratios, compressing data in tables and table fragments, consolidating free space (repacking), returning free space to a dbspace (shrinking), uncompressing data, and deleting individual table and fragment compression dictionaries.

An **admin()** command returns an integer; a **task()** command returns a string.

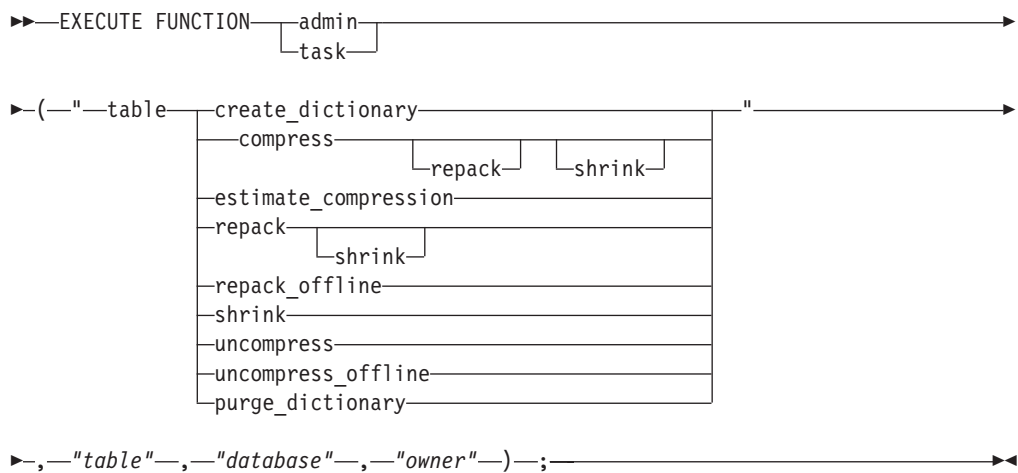
You cannot perform compression operations in indexes.

For information on the types of data that you can compress, compression ratios, compression estimates, and compression dictionaries, as well as procedures for using compression command parameters, see the *IBM Informix Administrator's Guide*. For information on utilities and the **sysmaster** table and view that display compression information, see "syscompdicts_full" on page 2-12.

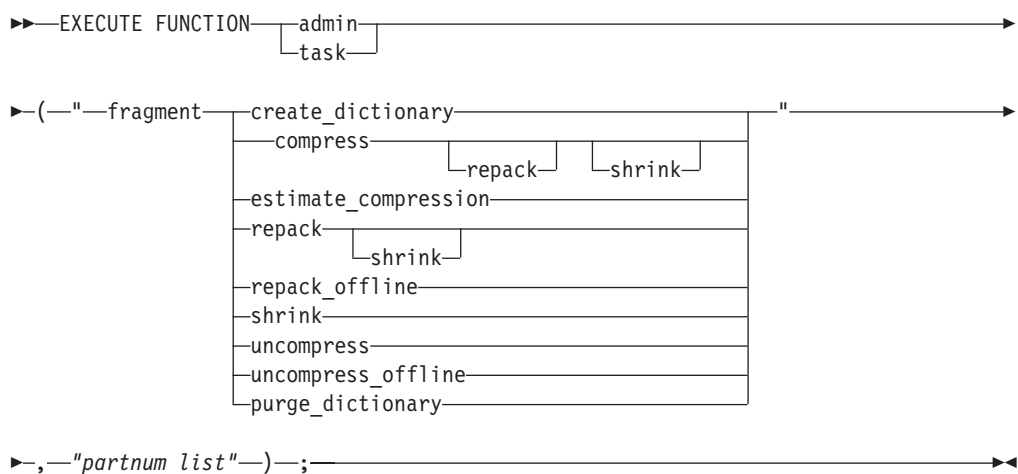
table or fragment arguments: Compress data and optimize storage (SQL administration API)

Call SQL administration API functions with **table** or **fragment** as the initial argument and one or more specific compression arguments to create compression dictionaries, estimate compression ratios, compress data in tables and table fragments, consolidate free space (repack), return free space to a dbspace (shrink), uncompress data, and delete compression dictionaries.

Syntax: Table Data Compression Command arguments



Syntax: Fragment Data Compression Command arguments



Command Arguments

The following table contains a brief explanation of each argument.

Table 21-20. Arguments for Compress and Uncompress Operations

Argument	Description
create_dictionary	<p>Builds a compression dictionary, which is a library of frequently occurring patterns and the symbol numbers that replace them in compressed rows.</p> <p>After a dictionary is created, any newly inserted or updated rows will be compressed if they are compressible. Existing rows are not compressed.</p>
estimate_compression	Estimates both a new compression ratio and a current ratio. The current ratio is 0.0 percent if the table is not compressed.
compress	<p>Compresses all existing rows in-place, without moving them (without repacking the table).</p> <p>If a compression dictionary for the target table or fragment does not exist, the compress operation also creates the dictionary.</p>
repack	<p>Consolidates free space by moving data to the front of the fragment or table.</p> <p>Because the repack operation moves rows while keeping the fragment online, other queries accessing the fragment that are using an isolation level below Repeatable Read might occasionally find the same row twice or miss finding a row. To avoid this possibility, use the Repeatable Read isolation level for concurrent queries; or, instead of using the repack argument, use the repack_offline argument.</p>
repack_offline	Consolidates free space by moving data to the front of the table or fragment, while holding an exclusive lock on the table or fragment. This operation prevents all other access to data until the operation is completed.
shrink	Returns free space at the end of a fragment or table to the dbspace, thus reducing the total size of the fragment or table.
uncompress	<p>Deactivates compression for new INSERT and UPDATE operations, uncompresses all compressed rows, and deactivates the compression dictionary. This operation also allocates new pages for a fragment and moves uncompressed rows that no longer fit on their original pages to the new pages.</p> <p>Because this operation moves rows while keeping the fragment online, other queries accessing the fragment that are using an isolation level below the Repeatable Read isolation level might occasionally find the same row twice or miss finding a row. To avoid this possibility, use the Repeatable Read isolation level for concurrent queries, or instead of using the uncompress argument, use the uncompress_offline argument.</p>
uncompress_offline	<p>Deactivates compression for new INSERT and UPDATE operations, uncompresses all compressed rows, and deactivates the compression dictionary, while holding an exclusive lock on the fragment. This prevents all other access to the fragment data until the operation is completed.</p> <p>This operation also allocates new pages for a fragment and moves uncompressed rows that no longer fit on their original pages to the new pages.</p>
purge_dictionary	Deletes an inactive compression dictionary, after you uncompress a table or fragment.
compress repack	Performs a combination of compress and repack operations.
compress repack shrink	Performs a combination of compress, repack, and shrink operations.
compress shrink	Performs a combination of compress and shrink operations.

Table 21-20. Arguments for Compress and Uncompress Operations (continued)

Argument	Description
repack shrink	Performs a combination of repack and shrink operations.

Command Elements

The following table shows the elements that you can use to create compression dictionaries, estimate compression ratios, compress data in tables and table fragments, repack space, return free space to a dbspace, uncompress data, and delete compression dictionaries.

Table 21-21. Compression and Storage Optimization Command Elements

Element	Description	Key Considerations
<i>database</i>	The name of the database that contains the table you want to compress.	Optional for a table. If you do not specify a <i>database</i> , Informix uses the current database. If you enter a database name, you must use the same uppercase or lowercase letters that are in system catalog tables.
<i>owner</i>	The name of the owner of the database that contains the table you want to compress.	Optional for a table. If you do not specify an <i>owner</i> , Informix uses the current owner. If you enter an owner name, you must use the same uppercase or lowercase letters that are in system catalog tables.
<i>partnum_list</i>	A space-separated list of partition numbers that belong to the same table.	Required for a fragment.
<i>table</i>	The name of the table that contains the data you want to compress.	Required for a table. You must use the same uppercase or lowercase letters that are in system catalog tables.

Usage

Informix uses the compression dictionary to compress data.

After you compress a table or fragment, Informix will automatically compress any new rows that you add to the table or fragment. If the data has changed significantly and you think that the compression dictionary might be built on older data, you should uncompress and then compress again.

The compress operation normally creates a quantity of free space on individual data and remainder pages, but the space is not consolidated at the end of the table or fragment. Instead, the space can be used to hold newly inserted rows, with the table not growing any larger until this space has been filled.

A compress operation, which only occurs online, compresses rows of a table in-place. The repack operation moves the rows. You can perform a repack operation online or offline. An online operation allows concurrent activity to occur on a table. However, this can result in *phantom rows*. (Phantom rows are rows that are initially modified or inserted during a transaction that is subsequently rolled back.)

To avoid phantom rows, you might want to repack offline, when you can afford to keep other users from accessing a table or fragment. For example, you could perform a compress operation with concurrent activity during the day, and then perform a repack_offline operation at night, when no concurrent activity is expected on the table.

You cannot perform an offline operation with an online operation. For example, while you can perform a combined compress repack operation, you cannot perform a combined compress repack_offline operation. If you want to repack offline, you must do this in two steps:

1. Perform a compress operation.
2. Perform a repack_offline operation.

Similarly you cannot perform a repack_offline shrink operation.

If light appends (unbuffered, unlogged insert operations) occur in a table or fragment while a repack operation is occurring, the repack operation will not complete the consolidation of space at the end of a table or fragment. The repack operation does not complete because the new extents are added in the location where the repack operation has already occurred, so space cannot be returned to the dbspace. To complete the repack process, you must run a second repack operation after light append activity has completed. This second repack operation builds on the work of the first repack operation.

The shrink operation is usually performed after a repack operation.

You can safely shrink the entire table without compromising the allocation strategy of the table. For example, if you have a fragmented table with one fragment for each day of the week and many fragments pre-allocated for future use, you can shrink the table without compromising this allocation strategy. If the table is empty, Informix shrinks the table to the initial extent size that was specified when the table was created.

Dropping or disabling indexes before you complete a repack_offline or uncompress_offline operation can decrease the amount of time that it takes the server to complete the operation. Afterwards, you can re-create or re-enable the indexes, preferably taking advantage of PDQ. Dropping or disabling the indexes and then creating or enabling them again can be faster than completing a repack_offline or uncompress_offline operation without doing this.

When you initiate a shrink operation, Informix shortens extents as follows:

- It shortens all extents except the first extent to as small a size as possible.
- If the table is entirely in the first extent (for example, because the table is an empty table), Informix does not shrink the first extent to a size that was smaller than the extent size that was specified when the table was created with the CREATE TABLE statement.

You can use the **MODIFY EXTENT SIZE** clause of the **ALTER TABLE** statement to reduce the current extent size. After you do this, you can rerun the shrink operation to shrink the first extent to the new extent size.

You cannot perform a compress, repack, repack_offline, shrink, uncompress, or uncompress_offline operation on a table or fragment while any of these operations is already occurring on the table or fragment.

Before you perform a **purge_dictionary** operation for tables and fragments, you must:

- Uncompress the tables and fragments.
When you uncompress a table or fragment, Informix marks the dictionary for the table or fragment as inactive.
- Be sure that Enterprise Replication functions do not need the compression dictionaries for older logs.
- Archive any dbspace that contains a table or fragment with a compression dictionary, even if you have uncompressed data in the table or fragment and the dictionary is no longer active.

The uncompress operation has no effect on any table or fragment it is applied to that is not compressed.

After you uncompress a table or fragment, you can perform a **purge_dictionary** operation to delete the dictionary for that table or fragment.

You can also delete all compression dictionaries or all compression dictionaries that were created before and on a specified date. For information, see “**purge compression dictionary arguments: Remove compression dictionaries (SQL administration API)**” on page 21-132.

You can cancel a command with a **compress** or **uncompress** argument, for example, by typing CTRL-C in DB-Access.

You can reissue commands with **repack**, **repack_offline**, **uncompress**, and **uncompress_offline** arguments after a prior interrupted command.

Compress, repack, and uncompress operations are logged, but run in small portions.

When you perform a combination of operations in a single command, the server runs the operations in this order:

```
create_dictionary compress repack shrink
```

If you change the fragmentation strategy for a table after you perform a compression operation, the table loses its compression status and will need to be recompressed.

Queries can access data in a compressed table.

Do not drop a dbspace that Change Data Capture (CDC) API is using, if the dbspace ever contained compressed tables, because this might delete compression dictionaries that CDC still needs.

Examples

The following command tells Informix to compress, repack, and shrink a table named **auto** in the **insurance** database of which **tjones** is the owner.

```
EXECUTE FUNCTION task("table compress repack shrink","auto",  
"insurance","tjones");
```

The following command tells Informix to uncompress the fragment with the partition number 14680071.

```
EXECUTE FUNCTION task("fragment uncompress","14680071");
```

The following command tells Informix to estimate the benefit of compressing a table named **home** in the **insurance** database of which **fgomez** is the owner.

```
EXECUTE FUNCTION task("table estimate_compression","home",  
"insurance","fgomez");
```

For more information on the output of the command, see “Output of the Estimate Compression Operation (SQL administration API).”

Output of the Estimate Compression Operation (SQL administration API)

After you run the command for estimating compression ratios, Informix displays information that shows the estimate of the compression ratio that can be achieved, along with the currently achieved compression ratio (if it exists).

Table 21-22. Information that an estimate_compression Operation Displays

Column	Information Displayed
est	This is the estimate of the compression ratio that can be achieved with a new compression dictionary. The estimate is a percentage of space saved compared to no compression.
curr	This is the estimate of the currently achieved compression ratio. This estimate is a percentage of space saved compared to no compression. 0.0% will always appear for non-compressed fragments or tables.
change	<p>This is the estimate of the percentage point gain (or possibly loss, although that should be rare) in the compression ratio that you could achieve by switching to a new compression dictionary. This is just the difference between est and curr.</p> <p>If the table or fragment is not compressed, you can create a compression dictionary with the compress parameter. If the fragment is compressed, you must perform an uncompress or uncompress_offline operation, before you can compress.</p>
partnum	This is the partition number of the fragment.
table	This is the full name of the table to which the fragment belongs, in format <i>database:owner.tablename</i> .

Example

The following output shows that a .4 percent increase in saved space can occur if you recompress the first fragment. A 75.7 percent increase can occur if you compress the second fragment, which is not compressed.

est	curr	change	partnum	table
75.7%	75.3%	+0.4	0x00200003	insurance:bwilson.disability
75.7%	0.0%	+75.7	0x00300002	insurance:pchang.workcomp

Output from compression estimates for tables and fragments look the same, except that the output for a table always shows all fragments in the table, while the output for a fragment only shows information for the specified fragments.

purge compression dictionary arguments: Remove compression dictionaries (SQL administration API)

Call the **admin()** or **task()** function with the **compression purge_dictionary** initial command to delete all inactive compression dictionaries or all inactive compression dictionaries that were created before a specified date. You must uncompress tables and fragments, which makes the dictionaries inactive, before you delete any compression dictionaries that were created for the tables and fragments.

Syntax: Compression Purge_Dictionary

```

--EXECUTE FUNCTION admin (task) ("compression purge_dictionary", "date") ;

```

Usage

Before you perform a **purge_dictionary** operation for tables and fragments, you must:

- Uncompress the tables and fragments.
When you uncompress a table or fragment, Informix marks the dictionary for the table or fragment as inactive.
- Be sure that Enterprise Replication functions do not need the compression dictionaries.
- Archive any dbspace that contains a table or fragment with a compression dictionary, even if you have uncompressed data in the table or fragment and the dictionary is no longer active.

The **compression purge_dictionary** command deletes all compression dictionaries.

The **compression purge_dictionary** command with a date as the second argument deletes all compression dictionaries that were created before and on a specified date. You can use any date in a format that can be converted to a DATE data type based on your locale and environment. For example, you can specify 03/29/2009, 03/29/09, or Mar 29, 2009.

You can also delete a specific compression dictionary by calling the **admin()** or **task()** function with **table** or **fragment** as the initial command and **purge_dictionary** as the next argument.

The following command tells Informix to remove all dictionaries that were created before and on July 8, 2009:

```
EXECUTE FUNCTION task("compression purge_dictionary", "07/08/2009");
```

The following command tells Informix to remove the inactive dictionary for a table named **auto** in the **insurance** database of which **tjones** is the owner.


```
EXECUTE FUNCTION task("table purge_dictionary",  
"auto", "insurance", "tjones");
```

Part 4. Appendixes

Appendix A. Files That the Database Server Uses

This appendix provides brief summaries of the files that you use when you configure and use the database server. It also includes descriptions of files (and one directory) created and used internally by the database server. For many of these files, your only responsibilities are to recognize that those files are legitimate and refrain from deleting them.

Pathnames that appear in the following format indicate files that reside on UNIX: **/directory/filename**. Pathnames that appear in the following format indicate files that reside on Windows: **\directory\filename**.

In some cases, environment variables are used to specify the initial pathname of a file. On UNIX, references to environment variables begin with a dollar sign: **\$INFORMIXDIR**. On Windows, references to environment variables begin and end with percent signs: **%INFORMIXDIR%**.

Database Server Files

Table A-1 lists the database server files and the directories in which they reside.

Table A-1. List of Files That the Database Server Uses

Filename	Directory	Purpose	Created
af.xxx	Specified by DUMPPDIR configuration parameter	Assertion-failure information	By the database server
ac_msg.log	/tmp (UNIX) %INFORMIXDIR%\etc (Windows)	archecker message log (for Technical Support)	By the database server
ac_config.std	\$INFORMIXDIR/etc (UNIX) %INFORMIXDIR%\etc (Windows)	Template for archecker-parameter values	By the database server
bar_act.log	/tmp (UNIX) %INFORMIXDIR%\etc (Windows)	ON-Bar activity log	By ON-Bar
bldutil.process_id	/tmp (UNIX) \tmp (Windows)	Error messages about the sysutils database appear in this file	By the database server
buildsmi.out (UNIX) buildsmi_out (Windows)	/tmp (UNIX) %INFORMIXDIR%\etc (Windows) %INFORMIXDIR%\etc\buildsmi_out. %INFORMIXSERVER% (Windows)	Error messages about SMI database	By the database server

Table A-1. List of Files That the Database Server Uses (continued)

Filename	Directory	Purpose	Created
concdr.sh	\$INFORMIXDIR /etc/conv (UNIX) %INFORMIXDIR%\etc\conv (Windows)	Converts the syscdr database to a format for an earlier version of the database server.	By the database server
.conf.dbservername		The onsnmp utility uses this file to obtain the database server configuration	By the database server
core	Directory from which the database server was invoked	Core dump	By the database server
Emergency boot files (For filenames, see “Emergency Boot Files for ON-Bar” on page A-5).	\$INFORMIXDIR/etc (UNIX) %INFORMIXDIR%\etc (Windows)	Used in a cold restore	By ON-Bar
gcore (UNIX)	Specified by DUMPDIR configuration parameter	Assertion failure information	By the database server
illsrar.xx	\$INFORMIXDIR/lib (UNIX) %INFORMIXDIR%\lib (Windows)	Shared libraries for the database server and some utilities	By install procedure
.informix (UNIX)	User's home directory	Set personal environment variables	By the user
informix.rc (UNIX)	\$INFORMIXDIR/etc	Set default environment variables for all users	By the database administrator
INFORMIXTMP	/INFORMIXTMP (UNIX) \%INFORMIXDIR%. (Windows)	Temporary directory for internal files	By the database server
.inf.servicename	/INFORMIXTMP (UNIX) drive:\INFORMIXTMP (Windows)	Connection information	By the database server
.infos.dbservername	\$INFORMIXDIR/etc (UNIX) %INFORMIXDIR%\etc (Windows)	Connection information	By the database server
.infxdirs	/INFORMIXTMP (UNIX) drive:\INFORMIXTMP (Windows)	Database server discovery file that onsnmp uses	By the database server
InstallServer.log (Windows)	C:\temp	Database server installation log	By the database server
ISM catalog	\$INFORMIXDIR/ism (UNIX) %ISMDIR% (Windows)	Records saved backup objects and storage volumes that IBM Informix Storage Manager (ISM) (ISM) uses	By ISM
ISM logs	\$INFORMIXDIR/ism/logs (UNIX) %ISMDIR%\logs (Windows)	Operator alert messages, backend status, additional ISM information	By ISM

Table A-1. List of Files That the Database Server Uses (continued)

Filename	Directory	Purpose	Created
ISMversion	\$INFORMIXDIR/ism (UNIX) %ISMDIR% (Windows)	ISM version	During installation
JVM_vpid	Specified by JVPLOG configuration parameter	Messages that the Java virtual machine generates	By the Java virtual machine
JVPLOG	Specified by JVPLOG configuration parameter	Messages from the Java virtual processor	By the database server
.jvpprops	Specified by JVPPROFILE configuration parameter	Template for Java VP properties	During installation
Message log	Specified by MSGPATH configuration parameter	Error messages and status information	By the database server
The ONCONFIG file	\$INFORMIXDIR/etc (UNIX) %INFORMIXDIR%\etc (Windows)	Configuration information	By the database administrator
onconfig	\$INFORMIXDIR/etc (UNIX) %INFORMIXDIR%\etc (Windows)	Default ONCONFIG file (optional)	By the database server administrator
onconfig.std	\$INFORMIXDIR/etc (UNIX)	Template for configuration- parameter values	During installation
oncfg_servername. servernum	\$INFORMIXDIR/etc (UNIX) %INFORMIXDIR%\etc (Windows)	Configuration information for whole-system restores	By the database server
onsnmp.servername	/tmp (UNIX) \tmp (Windows)	Log file that the onsnmp subagent uses	By onsnmp
onsrvapd.log	/tmp (UNIX) \tmp (Windows)	Log file for the database server daemon onsrvapd	By onsnmp
revcdr.sh	\$INFORMIXDIR /etc/conv (UNIX) %INFORMIXDIR% \etc\conv(Windows)	Reverts the syscdr database to an earlier format	By the database server
servicename.exp	/INFORMIXTMP (UNIX) drive:\INFORMIXTMP (Windows)	Connection information	By the database server
servicename.str	/INFORMIXTMP (UNIX) drive:\INFORMIXTMP (Windows)	Connection information	By the database server
shmem.xxx (UNIX)	Specified by DUMPDIR configuration parameter	Assertion-failure information	By the database server
sm_versions.std	\$INFORMIXDIR/etc (UNIX) %INFORMIXDIR%\etc (Windows)	Identifies storage manager in use	During installation

Table A-1. List of Files That the Database Server Uses (continued)

Filename	Directory	Purpose	Created
snmpd.log	/tmp (UNIX) \\tmp (Windows)	Log file for the SNMP master agent, snmpdm	By onsnmp
sqlhosts (UNIX)	\$INFORMIXDIR/etc	Connection information; contained in the registry on Windows	During installation; modified by the database server administrator
VP.servername.nnx	/INFORMIXTMP (UNIX) drive:\INFORMIXTMP (Windows)	Connection information	By the database server
xbsa.messages	\$INFORMIXDIR /ism/applogs (UNIX) %ISMDIR%\applogs (Windows)	XBSA library call information	By ISM

Descriptions of Files

This section provides short descriptions of the files listed in Table A-1 on page A-1.

af.xxx

The database server writes information about an assertion failure to the **af.xxx** file. The file is stored in the directory that the DUMPPDIR configuration parameter specifies. For more information, see the information on monitoring for data inconsistency in your *IBM Informix Administrator's Guide*.

ac_msg.log

When you use **archecker** with ON-Bar to verify a backup, it writes brief status and error messages to the ON-Bar activity log and writes detailed status and error messages to the **archecker** message log (**ac_msg.log**). Technical Support uses the **archecker** message log to diagnose problems with backups and restores.

You specify the location of the **archecker** message log with the AC_MSGPATH configuration parameter. For more information, see the *IBM Informix Backup and Restore Guide*.

ac_config.std

The **ac_config.std** file contains the default **archecker** (archive checking) utility parameters. To use the template, copy it into another file, and modify the values. For a comprehensive list of the **archecker** parameters and how to use **archecker** with ON-Bar, see the *IBM Informix Backup and Restore Guide*.

bar_act.log

As ON-Bar backs up and restores data, it writes progress messages, warnings, and error messages to the ON-Bar activity log (**bar_act.log**). You specify the location of the ON-Bar activity log with the BAR_ACT_LOG configuration parameter. For more information, see the *IBM Informix Backup and Restore Guide*.

bldutil.process_id

If the database server cannot build the sysutils database, it creates the **bldutil.<process_id>** file which contains the error messages. The **process_id** value is the process ID of the **bldutil.sh** program. To access this output file, specify **\${RESFILE}**.

buildsmi.out (UNIX) or buildsmi_out (Windows)

If the database server cannot build the **sysmaster** database, it places a message in the message log that refers you to the output file of the **buildsmi** script, whose filename and pathname are platform-dependent.

On UNIX, the file specification is

/tmp/buildsmi.out

On Windows, the file specification is

%INFORMIXDIR%\etc\buildsmi_out.%INFORMIXSERVER%

Here **%INFORMIXSERVER%** is the name of the Informix instance. This file provides information about why the build failed. For information about the **sysmaster** database, refer to Chapter 2, “The sysmaster Database.”

concdr.sh

To convert the **syscdr** database from 7.31, 9.20, 9.21, 9.3 or 9.4 to 10.0 format, run the **concdr.sh** script on UNIX or the **concdr.bat** script on Windows. For details, see the *IBM Informix Migration Guide*.

.conf.dbservername

The **.conf.dbservername** file is created when you initialize the database server. The **onsnmp** utility queries this file to find out the configuration status of the database server. Do not delete this file.

The **.conf.dbservername** file contains information on shared memory and configuration that allows shared-memory clients to connect to the database server when they use utilities such as **onstat** or **onmode**.

core

The **core** file contains a core dump caused by an assertion failure. The database server writes this file to the directory from which the database server was invoked. For more information on monitoring for data inconsistency, see the chapter on consistency checking in the *IBM Informix Administrator's Guide*.

Emergency Boot Files for ON-Bar

The ON-Bar emergency boot files contain the information needed to perform a cold restore, and are updated after every backup. For details, see the *IBM Informix Backup and Restore Guide*.

The filename for the Informix emergency boot file is **ixbar.servernum**.

gcore.xxx (UNIX)

The database server writes information about an assertion failure to the **gcore.xxx** file. The file is stored in the directory specified by the DUMPDIR configuration parameter. For more information on monitoring for data inconsistency, see the chapter on consistency checking in the *IBM Informix Administrator's Guide*.

illsrta.xx

The **illsrta.xx** files are shared libraries that the database server and some database server utilities use. The shared libraries, if supported on your platform, are installed in **\$INFORMIXDIR/lib** or **%INFORMIXDIR%\lib**.

The naming convention of the Informix shared library filename is as follows:
illsrta.xx

<i>lll</i>	library class (for example, asf or smd)
<i>s</i>	library subclass (d=DSA; s=standard)
<i>rr</i>	major release number (for example, 07 or 08)
<i>a</i>	library version ID (for example, a or b)
<i>xx</i>	shared-library filename extension (for example, so)

UNIX Only:

Symbolic links to these files are automatically created in **/usr/lib** when the products are installed on your computer. The symbolic links to the shared libraries in **/usr/lib** are automatically created by the product installation procedures. However, if your **\$INFORMIXDIR** is not installed using the standard installation method (for example, your **\$INFORMIXDIR** is NFS-mounted from another computer), you or your system administrator might need to create manually the symbolic links of the shared libraries in **/usr/lib**.

~/.informix

The **~/.informix** file is the *private-environment file*. Users can create this file and store it in their home directory. The *IBM Informix Guide to SQL: Reference* discusses the environment-configuration files.

informix.rc (UNIX)

The **/informix.rc** file is the *environment-configuration file*. You can use it to set environment variables for all users of IBM Informix products. The *IBM Informix Guide to SQL: Reference* discusses the environment-configuration files.

INFORMIXTMP

The **INFORMIXTMP** directory is an *internal database server directory*. During initialization, the database server creates this directory (if it does not exist yet) for storing internal files that must be local and relatively safe from deletion. The **onsnmp** utility uses the files in the **INFORMIXTMP** directory.

.inf.servicename

The database server creates the **.inf.servicename** file if any DBSERVERNAME or DBSERVERALIASES uses a shared-memory connection type. The database server removes the file when you take the database server offline. The name of this file is derived from the servicename field of the **sqlhosts** file or registry.

The database server keeps information about client/server connections in this file. You do not use the **.inf.servicename** file directly. You only need to recognize that it is a legitimate file when it appears in the **INFORMIXTMP** directory.

If this file is accidentally deleted, you must restart the database server.

.infos.dbservername

The database server creates the **.infos.dbservername** file when you initialize shared memory and removes the file when you take the database server offline. This file resides in **\$INFORMIXDIR/etc** or **%INFORMIXDIR%\etc**. The name of this file is derived from the DBSERVERNAME parameter in the ONCONFIG configuration file.

The **.infos.dbservername** file contains information on shared memory and configuration that allows shared-memory clients to connect to the database server when they use utilities such as **onstat** or **onmode**. Do not delete this file.

.infxdirs

The database server maintains an **.infxdirs** file in the **INFORMIXTMP** directory. This file contains a line for every **INFORMIXDIR** from which a database server has been launched. If you remove the **.infxdirs** file, **onsnmp** cannot discover any database servers until the next time you restart the database server. Each time you restart the database server, it re-creates the **.infxdirs** file.

InstallServer.log (Windows)

The database server creates the **InstallServer.log** during installation.

ISM Catalog

ISM creates the ISM catalog during the **ism_startup** initialization. The ISM catalog records information about backup and restore save sets and about storage volumes that the storage manager uses. The ISM catalog records are stored in the **mm**, **index**, and **res** files in the **\$INFORMIXDIR/ism** or **%ISMDIR%\ism** directory. For more information, see the *IBM Informix Storage Manager Administrator's Guide*.

ISM Logs

ISM creates several logs during ON-Bar backup and restore operations. The message window in the ISM Administrator GUI displays messages from these logs.

Log	Description
------------	--------------------

daemon.log	ISM backend status
-------------------	--------------------

messages

Operator alert messages

summary

Additional ISM information

For more information, see the *IBM Informix Storage Manager Administrator's Guide*.

ISMversion

The **ISM version** file, which is installed with the database server, identifies the ISM version. Do not edit this file.

JVM_vpid file

The JVM_vpid file is a log file, where vp_id is the process ID of the Java virtual processor. For more information, see the *J/Foundation Developer's Guide*.

JVPLOG

When JVPDEBUG is set to 1, the database server writes tracing messages to the **JVPLOG** file. You can adjust the tracing level. On UNIX, you can have multiple **JVPLOG** files, one for each JVP virtual processor. On Windows, only one **JVPLOG** file exists. To obtain the JVP IDs, use the **onstat -g glo** command. For more information, see *J/Foundation Developer's Guide*.

.jvpprops

The **.jvpprops** file sets the Java virtual processor properties. Copy the **.jvpprops.template** to a new file named **.jvpprops**, and modify the values. For more information, see *J/Foundation Developer's Guide*.

Message Log

The database server writes status and error information to the message-log file. You specify the filename and location of the message log with the MSGPATH configuration parameter. For more information, refer to “MSGPATH configuration parameter” on page 1-96.

onconfig.std

The **onconfig.std** file serves as the template for creating the ONCONFIG configuration file. To use the template, copy it to another file and modify the values.

Important: Do not modify or delete the **onconfig.std** file. If you omit a parameter value in your copy of the configuration file, the database server either uses default values in **onconfig.std** template file or calculates values based on other parameter values.

Related reference:

“onconfig Portal: Configuration parameters by functional category” on page 1-3

The ONCONFIG File

The *current configuration file* is the `%INFORMIXDIR%\etc\%ONCONFIG%` or `$INFORMIXDIR/etc/$ONCONFIG` file. The database server uses the ONCONFIG file during initialization.

If you start the database server with `oninit` and do not explicitly set the **ONCONFIG** environment variable, the database server looks for configuration values in the **onconfig.std** file. If no **onconfig.std** file exists, the database server returns the following error message:

WARNING: Cannot access configuration file \$INFORMIXDIR/etc/\$ONCONFIG.

For more information on the order of files where the database server looks for configuration values during initialization, refer the material on initializing the database server in the *IBM Informix Administrator's Guide*.

For more information on setting up your ONCONFIG file, refer to the materials on installing and configuring the database server in the *IBM Informix Administrator's Guide*.

onconfig

The **onconfig** file is an optional file that you create in the `$INFORMIXDIR/etc` or `%INFORMIXDIR%\etc` directory. The **onconfig** file is the default configuration file if the **ONCONFIG** environment variable is not set. For more information, refer to processing the configuration file in the *IBM Informix Administrator's Guide*.

To create the **onconfig** file, you can copy **onconfig.std** or one of your customized configuration files. For more information on setting up your **ONCONFIG** file, refer to installing and configuring the database server in the *IBM Informix Administrator's Guide*.

oncfg_servername.servernum

The database server creates the **oncfg_servername.servernum** file in the `$INFORMIXDIR/etc` or `%INFORMIXDIR%\etc` directory when you initialize disk space. The database server updates the file every time that you add or delete a dbspace, a logical-log file, or a chunk. The database server uses the **oncfg_servername.servernum** file when it salvages logical-log files during a whole-system restore. The database server derives the name of this file from the values of the DBSERVERNAME and SERVERNUM parameters in the ONCONFIG configuration file.

The database server uses the **oncfg_servername.servernum** files, so do not delete them. For more information, refer to creating the **oncfg_servername.servernum** file in the *IBM Informix Administrator's Guide* and the *IBM Informix Backup and Restore Guide*.

onsnmp.servername

The **onsnmp** subagent uses this log file. For more information, see the *IBM Informix SNMP Subagent Guide*.

This log file is called **onsnmp.servername** on Informix.

onsrvapd.log

The **onsrvapd** daemon uses this log file. For more information, see the *IBM Informix SNMP Subagent Guide*.

revcdr.sh

To revert the **syscdr** database from 10.0 to 9.4, 9.3, 7.31, 9.20, or 9.21 format, run the **revcdr.sh** script on UNIX or the **revcdr.bat** script on Windows. For details, see the *IBM Informix Migration Guide*.

shmem.xxx (UNIX)

The database server writes information about an assertion failure to the **shmem.xxx** file. The file is stored in the directory that the DUMPDIR configuration parameter specifies. For more information on monitoring for data inconsistency, see the chapter on consistency checking in the *IBM Informix Administrator's Guide*.

sm_versions.std

The **sm_versions.std** file is a template for the **sm_versions** file that you create. The **sm_versions** file contains a line identifying the current storage-manager version.

The storage manager uses the data in the **sm_versions** file (no **.std** suffix). To update the storage-manager version, edit the **sm_versions** file and then run the **ism_startup** command. For more information, see the *IBM Informix Backup and Restore Guide*.

snmpd.log

The SNMP master agent, **snmpdm** uses this log file. For more information, see the *IBM Informix SNMP Subagent Guide*.

sqlhosts

UNIX Only:

The **sqlhosts** file is the *connectivity file* on UNIX platforms. It contains information that lets an IBM Informix client connect to an IBM Informix database server. For more information on the **sqlhosts** file, see client/server communications in the *IBM Informix Administrator's Guide*.

Windows Only:

On Windows, the connectivity information is in the **HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS** key in the Windows registry.

VP.servername.nnx

The database server creates the **VP.servername.nnx** file, if needed, when you initialize shared memory. The name of this file comes from DBSERVERNAME or DBSERVERALIASES in the ONCONFIG file, the VP number (**nn**), and an internal identifier (**x**).

The database server keeps information about client/server connections in the **VP.servername.nnx** file. You do not use the file directly. You only need to recognize that it is a legitimate file.

If this file is accidentally deleted, you must restart the database server.

xbsa.messages

The **xbsa.messages** log contains XBSA library call information. ON-Bar and ISM use XBSA to communicate with each other. Technical Support would use the **xbsa.messages** log to diagnose problems with ON-Bar and ISM communications.

Appendix B. Troubleshooting errors

Occasionally, a series of events causes the database server to return unexpected error codes.

You can use the following diagnostic tools to gather information for troubleshooting errors:

- **onmode -I**
- tracepoints
- The **ifxcollect** tool

Collecting Diagnostics using onmode -I

To help collect additional diagnostics, you can use **onmode -I** to instruct the database server to perform the diagnostics collection procedures that the *IBM Informix Administrator's Guide* describes. To use **onmode -I** when you encounter an error number, supply the *iserrno* and an optional session ID. For more information about **onmode**, see Chapter 14, "The onmode utility," on page 14-1.

Creating Tracepoints

Tracepoints are useful in debugging user-defined routines written in C. You can create a user-defined tracepoint to send special information about the current execution state of a user-defined routine.

Each tracepoint has the following parts:

- A *trace* groups related tracepoints together so that they can be turned on or off at the same time.

You can either use the built-in trace called **_myErrors** or create your own. To create your own trace, you insert rows into the **systracees** system catalog table.

- A *trace message* is the text that the database server sends to the tracing-output file.

You can store internationalized trace messages in the **systracemsgs** system catalog table.

- A *tracepoint threshold* determines when the tracepoint executes.

By default, the database server puts all trace messages in the trace-output file in the **tmp** directory with the following filename:

`session_num.trc`

For more information on tracing user-defined routines, see the *IBM Informix DataBlade API Programmer's Guide*.

Collecting data with the ifxcollect tool

You can use the **ifxcollect** tool to collect diagnostic data if necessary for troubleshooting a specific problem, such as an assertion failure. You can also specify options for transmitting the collected data via the File Transfer Protocol (FTP). .

The **ifxcollect** tool resides in the **\$NFORMIXDIR/bin** directory. Output files that **ifxcollect** commands generate reside in the **\$NFORMIXDIR/isa/data** directory.

Syntax

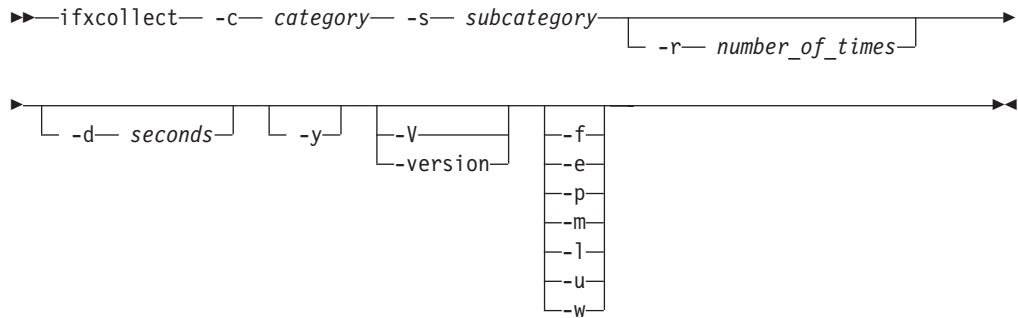


Table B-1. Options for data collection

Element	Description	Key Considerations
-c category	Tells the server to collect data in the specified category.	You must specify the category of data to collect.
-s category	Tells the server to collect data in the specified subcategory.	You must specify the subcategory of data to collect.
-r number of times	Specifies the number of times to repeat data collection	Optional.
-d number of seconds	Specifies the number of times to pause between collection operations	Optional.
-y	Causes the database server to automatically respond yes to all prompts.	Optional.
-V	Displays the software version number and the serial number	Optional. See "Obtaining Utility Version Information" on page 6-1
-version	Displays the build version, host, OS, number and date, as well as the GLS version	Optional. See "Obtaining Utility Version Information" on page 6-1

Table B-2. FTP options if you also transmitting data

Element	Description	Key Considerations
-f	FTP the entire collection	Required for the transmission of data.
-e email address	E-mail address	Required for the transmission of data.
-p the PMR number	PMR number	Required for the transmission of data.
-m machine name	Machine to connect to	Required for the transmission of data.
-l directory	Directory that contains the data	Required for the transmission of data.

Table B-2. FTP options if you also transmitting data (continued)

Element	Description	Key Considerations
-u <i>user name</i>	User name for the FTP	Required for the transmission of data.
-w <i>password</i>	Password for the FTP	Required for the transmission of data.

Usage

The following table shows the combination of categories and subcategories that you can use in your commands.

Table B-3. Category and subcategory combinations

Category and subcategory	Explanation
-c <i>ids</i> -s <i>general</i>	Collects general data for issues related to all Informix products
-c <i>af</i> -s <i>general</i>	Collects general data for assertion failures
-c <i>er</i> -s <i>general</i>	Collects general data for Enterprise Replication
-c <i>er</i> -s <i>init</i>	Collects general data for Enterprise Replication initialization issues
-c <i>performance</i> -s <i>general</i>	Collects data for performance issues
-c <i>performance</i> -s <i>cpu</i>	Collects data for CPU utilization issues
-c <i>onbar</i> -s <i>archive_failure</i>	Collects data for onbar archive failures
-c <i>onbar</i> -s <i>restore_failure</i>	Collects data for onbar restore failures
-c <i>ontape</i> -s <i>archive_failure</i>	Collects data for ontape archive failures
-c <i>ontape</i> -s <i>restore_failure</i>	Collects data for ontape restore failures
-c <i>connection</i> -s <i>failure</i>	Collects data for connection failures
-c <i>connection</i> -s <i>hang</i>	Collects data for connection hangs
-c <i>cust</i> -s <i>prof</i>	Collects customer profile information

To view all **ifxcollect** utility command options, type **ifxcollect** at the command prompt.

Examples

To collect information for a general assertion failure, run this command:

```
ifxcollect -c af -s general
```

To collect information for a performance problem related to CPU utilization, run this command:

```
ifxcollect -c performance -s cpu
```

To include FTP information, specify the additional information as shown in this example:

```
-f -e user_name@company_name.org -p 9999.999.999
-f -m machine -l /tmp -u user_name -w password
```

Appendix C. Event Alarms

The database server provides a mechanism for automatically triggering administrative actions based on an event that occurs in the database server environment. This mechanism is the event-alarm feature. Events can be informative (for example, Backup Complete) or can indicate an error condition that requires your attention (for example, Unable to Allocate Memory).

Related reference:

“ALARMPROGRAM configuration parameter” on page 1-27

“ALRM_ALL_EVENTS configuration parameter” on page 1-29

Using ALARMPROGRAM to Capture Events

On UNIX, use the **alarmprogram.sh** and on Windows, use the **alarmprogram.bat** shell script, for handling event alarms and starting automatic log backups. For the setup instructions, see “ALARMPROGRAM configuration parameter” on page 1-27.

To automate logical-log backups only, two ready-made scripts are provided: **log_full.[sh | bat]** and **no_log.[sh | bat]**. Set ALARMPROGRAM to the full path name of the script. For information, see “ALARMPROGRAM configuration parameter” on page 1-27.

Setting ALRM_ALL_EVENTS

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

Writing Your Own Alarm Script

Alternatively, you can write your own shell script, batch file, or binary program that contains the event-alarm parameters. When an event occurs, the database server invokes this executable file and passes it the event-alarm parameters (see Table C-1 on page C-4). For example, your script can use the **_id** and **_msg** parameters to take administrative action when a table failure occurs. Set ALARMPROGRAM to the full pathname of this executable file.

Customizing the ALARMPROGRAM Scripts

Follow these steps to customize the **alarmprogram.[sh | bat]** script. You can use **alarmprogram.[sh | bat]** instead of **log_full.[sh | bat]** to automate log backups.

To customize the ALARMPROGRAM scripts:

1. Change the value of ADMINMAIL to the email address of the database server administrator.
2. Change the value of PAGERMAIL to the pager service email address.
3. Set the value of the parameter MAILUTILITY with **/usr/bin/mail** for UNIX and **\$INFORMIXDIR/bin/ntmail.exe** for Windows.

4. To automatically back up logical logs as they fill, change BACKUP to yes. To stop automatic log backups, change BACKUP to any value other than yes.
5. In the ONCONFIG file, set ALARMPROGRAM to the full pathname of **alarmprogram.[sh | bat]**.
6. Restart the database server.

Alarms with a severity of 1 or 2 do not write any messages to the message log nor send email. Alarms with severity of 3 or greater send email to the database administrator. Alarms with severity of 4 and 5 also notify a pager via email.

Precautions for Foreground Operations in Alarm Scripts

To ensure continuous server availability, do not run certain foreground operations in an alarm script.

When the server invokes an alarm script, the server sometimes waits for the script to complete before proceeding. For example:

- When an alarm is invoked because of a fatal error, the server waits for the script to finish writing information to the error log. In certain situations, alarm events 5 and 6 are run in the foreground.
- Some Enterprise Replication event alarms run in the foreground, such as event alarms 31, 34, 37, and 39.

Because the server might need to wait for the alarm program script to finish, do not run the following operations in the foreground in an alarm script:

- An onmode command that forces user connections off the server such as onmode -u or onmode -yuk. These kinds of onmode commands can cause a deadlock between the server and the alarm script because the server might wait for the alarm script to complete while the alarm script that executed the onmode command waits for the user sessions to shut down, and one of those sessions is running the alarm script itself.
- Operations that might take a long time to complete or that have a highly variable run time. Operations that take a long time to complete can cause the server to appear as if it is not responding while the operation is running.

If you need to run the above operations in an alarm script, run them in the background using one of the following operating system utilities:

On UNIX: Use the nohup utility. For example, nohup onmode -yuk & instructs nohup to continue running the command even if its parent terminates and the ampersand, &, runs the command in the background so it will not block execution of the alarm program script itself.

On Windows: Use the start utility with the /B flag. For example, start /B onmode -yuk.

Interpreting event alarm messages

Some of the events that the database server reports to the message log cause it to invoke the alarm program. The class messages indicate the events that the database server reports.

The database server reports a nonzero exit code in the message log. In the alarm program, set the EXIT_STATUS variable to 0 for successful completion and to another number for a failure.

For example, if a thread attempts to acquire a lock, but the maximum number of locks that LOCKS specifies has already been reached, the database server writes the following message to the message log:

```
10:37:22 Checkpoint Completed: duration was 0 seconds.
10:51:08 Lock table overflow - user id 30032, rstcb 10132264
10:51:10 Lock table overflow - user id 30032, rstcb 10132264
10:51:12 Checkpoint Completed: duration was 1 seconds.
```

When the database server invokes the **alarmprogram.[sh|bat]** program or your alarm program, the database server generates a message that describes the severity and class of the event. If the severity is greater than 2, the message takes the following format:

Action	Message
A reasonably severe server event	Severity: 3 Class ID: 21 Class msg: Database server resource overflow: 'Locks'. Specific msg: Lock table overflow - user id 30032, rstcb 10132264 See Also: # optional message Event ID: 21005
The message that appears at the end of each e-mailed message	This e-mail was generated by the server ALARMPROGRAM script on <i>servername</i> because something untoward just happened to <i>eventname</i> .

Events in the ph_alert Table

All event alarms that are generated are inserted in the **ph_alert** table in the **sysadmin** database.

You can query the **ph_alert** table on local or remote server to view the recent event alarms for that server. You can write SQL scripts based on the **ph_alert** table to handle event alarms instead of using the scripts controlled by the ALARMPROGRAM configuration parameter.

By default, alerts remain in the **ph_alert** table for 15 days before being purged.

Example

The following example shows an event alarm in the **ph_alert** table:

```
SELECT * FROM ph_alerts WHERE alert_object_type=ALARM;
```

```
id                34
alert_task_id     18
alert_task_seq    10
alert_type        INFO
alert_color       YELLOW
alert_time        2010-03-08 12:05:48
alert_state       NEW
alert_state_chang+ 2010-03-08 12:05:48
alert_object_type ALARM
alert_object_name 23
alert_message     Logical Log 12 Complete, timestamp: 0x8e6a1.
alert_action_dbs  sysadmin
alert_action
alert_object_info 23001
```

Related reference:

“The ph_alert Table” on page 3-5

“Event Alarm Parameters”

Event Alarm Parameters

Event alarms have five parameters that describe each event.

The following table lists the parameters that are part of event alarm.

Table C-1. Event Alarm Parameters

Parameter	Description	Data Type
severity	The severity of the event.	integer
class_id	A numeric identifier that classifies the type of event that has occurred.	integer
class_msg	A brief messages that describes the classification of the event.	string
specific_msg	Specific messages that describes the event that occurred.	string
see_also	A reference to a file that contains additional information about the event.	string
uniqueid	A unique event identifier for the specific message.	bigint

Event Severity

An event severity code is a numeric indication of the seriousness of an event. Every event that is included in the message log contains a severity code. The event severity code is the first parameter that is sent to the alarm program. In the **ph_alert** table, the event severity is reflected by a combination of the alert color and the alert type. The event severity codes are listed in the following table.

Table C-2. Event Severity Codes

Severity	Description
1	Not noteworthy. The event (for example, date change in the message log) is not reported to the alarm program unless ALRM_ALL_EVENTS configuration parameter is enabled. In the ph_alert table, the alert color is GREEN and the alert type is INFO.
2	Information. No error has occurred, but some routine event completed successfully (for example, checkpoint or log backup completed). In the ph_alert table, the alert color is YELLOW and the alert type is INFO.
3	Attention. This event does not compromise data or prevent the use of the system; however, the event warrants your attention. For example, one chunk of a mirrored pair goes down. An email is sent to the system administrator. In the ph_alert table, the alert color is YELLOW and the alert type is WARNING.
4	Emergency. Something unexpected occurred that might compromise data or access to data. For example an assertion failure, or oncheck reports data corrupt. Take action immediately. The system administrator is paged when this event severity occurs. In the ph_alert table, the alert color is RED and the alert type is ERROR.

Table C-2. Event Severity Codes (continued)

Severity	Description
5	Fatal. Something unexpected occurred and caused the database server to fail. The system administrator is paged when this event severity occurs. In the ph_alert table, the alert color is RED and the alert type is ERROR.

Class ID

The class ID is an integer that identifies the event that causes the database server to run your alarm program. The class ID is the second parameter that the database server displays in your alarm program.

The class ID is stored in the **alert_object_name** column in the **ph_alert** table.

Class Message

The class message is a text message briefly describes, or classifies, the event that causes the database server to run your alarm program. The class messages is the third parameter that the database server displays in your alarm program.

Specific Message

The specific message is a text messages the describes in more detail the event that causes the database server to run your alarm program. The specific message is the fourth parameter that the database server displays in your alarm program. For many alarms, the text of this message is the same as the message that is written to the message log for the event.

The specific message is stored in the **alert_message** column in the **ph_alert** table.

See Also Paths

For some events, the database server writes additional information to a file when the event occurs. The path name in this context refers to the path name of the file where the database server writes the additional information.

Event ID

The event ID is a unique number for each specific message. You can use the event ID in custom alarm handling scripts to create responses to specific events.

The event ID is stored in the **alert_object_info** column in the **ph_alert** table.

Related concepts:

“Events in the ph_alert Table” on page C-3

Event alarm IDs

The class ID for event alarms indicates the type of event. The event ID indicates the specific event.

The following table lists event alarm IDs and messages or where to find more information. Alarms with a severity of 4 and 5 have additional explanations and

user actions. Many of the issues that trigger event alarms also result in messages in the online message log. The location of the message log is specified by the MSGPATH configuration parameter.

Table C-3. Event Alarms

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1001	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: Page allocation error on 'object'	
Class ID: 1 Event ID: 1002	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: Row allocation error on 'object'	
Class ID: 1 Event ID: 1003	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: Slot allocation error for 'object'	
Class ID: 1 Event ID: 1004	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error prevented the database server to find the next possible data page in this tblspace.	
Class ID: 1 Event ID: 1005	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: Dropping wrong TBLSpace, requested tblspace_name != actual tblspace_name	
Class ID: 1 Event ID: 1006	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error which may have been caused due to data corruption prevented the database server from altering the bitmap pages for this partition.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1007	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: [3] An internal error which may have been caused due to corrupted bitmap pages as the database server is still in the process of converting them.	
Class ID: 1 Event ID: 1008	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error which may have been caused due to unconverted bitmap pages.	
Class ID: 1 Event ID: 1009	4	Class message: Table failure: 'dbname:"owner".tablename' Specific message: Page Check Error in <i>object</i>	The database server detected inconsistencies while checking a page that was being read into internal buffers. Online log: Assertion failure or assertion warning with a description of the problem. Server state: Online or offline, depending on how serious the problem is. User action: Follow the suggestions in the online log. Typically, run the oncheck -cD command on the table mentioned in the class message or on the database.
Class ID: 1 Event ID: 1010	4	Class message: Table failure: 'dbname:"owner".tablename' Specific message: Bad rowid <i>rowid</i>	The database server detected an invalid row ID. Online log: Assertion warning with a description of where the problem was found. Server state: Online. User action: Repair the index by running the oncheck -cI command on the table mentioned in the class message or on the database.
Class ID: 1 Event ID: 1011	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: Closing TBLSpace <i>tblspace_name</i>	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1012	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: Cannot recreate index <i>index_name</i> for partnum <i>partition_number</i> , iserrno = <i>error_number</i>	
Class ID: 1 Event ID: 1013	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to initialize the type of set read operation.	
Class ID: 1 Event ID: 1014	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to read records from the tblspace's pages	
Class ID: 1 Event ID: 1015	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to read the current record.	
Class ID: 1 Event ID: 1016	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to initialize the set read buffer.	
Class ID: 1 Event ID: 1017	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to set the new mode on the bitmap page.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1018	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was attempting to convert bitmap pages to the correct format.	
Class ID: 1 Event ID: 1019	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to modify the bitmap pages during light append operation.	
Class ID: 1 Event ID: 1020	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to perform light scan operation.	
Class ID: 1 Event ID: 1021	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error occurred while the database server was trying to perform light scan I/O operation.	
Class ID: 1 Event ID: 1022	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to validate light append buffer.	
Class ID: 1 Event ID: 1023	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to write the next record to the page in the light append buffer.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1024	4	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it tried to open a light append for a tblspace.	The database server encountered an internal error during a light append operation on a tblspace. Online log: Assertion failure. Server state: Online. User action: Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 1 Event ID: 1025	4	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it tried to load the first bitmap page for a light append operation.	The database server encountered an internal error during a light append operation. Online log: Assertion Failure : Light Append(Redo/Undo): Can't find bitmap page Server state: Online. User action: Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 1 Event ID: 1026	4	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it tried to write the cached bitmap pages for a light append operation.	The database server encountered an internal error during a light append operation. Online log: Assertion failure. Server state: Online. User action: Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 1 Event ID: 1027	2	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal deadlock condition was caught by the Lock Manager in the database server.	
Class ID: 1 Event ID: 1028	2	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal deadlock condition was caught by the Lock Manager in the database server.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1029	4	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to map the logical page number in the tblspace to its physical location in the chunk.	The database server could not access a table because of an inconsistency between the physical page and its logical page number. Online log: Assertion failure with the page information. Server state: Online. User action: Run the oncheck -cDI command on the table mentioned in the class message or on the database, fix any issues reported, and then try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 1 Event ID: 1030	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to allocate the alter information.	
Class ID: 1 Event ID: 1031	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to prepare the list of operations to be performed on a compressed row.	
Class ID: 1 Event ID: 1032	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to insert an operation in the list of operations based on the offset in the row where the new operation points to.	
Class ID: 1 Event ID: 1033	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it detected an inconsistency with the operation list.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1034	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it tried to free the partition header page.	
Class ID: 1 Event ID: 1035	3 or 4	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it tried to validate the partition header page.	The database server cannot access a table because of a validation error for the tblspace page. Online log: Assertion with details about the table. Server state: Online. User action: Examine the online log for information about the specified table. Run the oncheck -pt command on the table or on the database and correct any errors found. Retry the original operation. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 1 Event ID: 1036	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it tried to update the special columns list during an alter table command processing.	
Class ID: 1 Event ID: 1037	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it tried to log the completion of the alter and remove the associated version information from the tblspace's header page.	
Class ID: 1 Event ID: 1038	3	Class message: Table failure: 'dbname:"owner".tabname' Specific message: An internal error was reported by the database server when it detected a buffer inconsistency.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1039	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to construct a forwarded row into a single tuple.	
Class ID: 1 Event ID: 1040	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to read the data from a partition into the set read buffer.	
Class ID: 1 Event ID: 1041	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to read the data row for a given rowid.	
Class ID: 1 Event ID: 1042	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to alter the row in memory to the latest schema.	
Class ID: 1 Event ID: 1043	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to undo the alter of a bitmap page.	
Class ID: 1 Event ID: 1044	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to undo the addition of special column descriptors from the tblspace's header page.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 1 Event ID: 1045	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to undo the addition of the new version to a partition.	
Class ID: 1 Event ID: 1046	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to allocate the file descriptor for a partnum.	
Class ID: 1 Event ID: 1047	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: An internal error was reported by the database server when it tried to free the file descriptor for a partnum.	
Class ID: 1 Event ID: 1048	3	Class message: Table failure: 'dbname:"owner".tablename' Specific message: Error updating table record.	
Class ID: 2 Event ID: 2001	3 or 4	Class message: Index failure: 'dbname:"owner".tablename:idxname' Specific message: Frigid <i>fragment_id</i> , Rowid <i>rowid</i> not found for delete in partnum <i>partition_number</i>	A DELETE operation failed because the database server could not find the record to be deleted in an index. Online log: Assertion indicating that a DELETE operation failed and details of the table and index where the problem occurred. Server state: Online. User action: Run the oncheck -cI command on the specified table and index, or on the database, and correct any errors found. Retry the original operation. If the operation fails again, note all circumstances and contact IBM Software Support.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 2 Event ID: 2002	3	Class message: Index failure: ' <i>dbname:"owner".tabname:idxname</i> ' Specific message: An internal error was raised due to an inconsistency in the index which is preventing the database server to position on the first record in that index.	
Class ID: 2 Event ID: 2003	3	Class message: Index failure: ' <i>dbname:"owner".tabname:idxname</i> ' Specific message: An internal error was raised due to an inconsistency in the index which is preventing the database server to read ahead pages in that index.	
Class ID: 2 Event ID: 2004	4	Class message: Index failure: ' <i>dbname:"owner".tabname-idxname</i> ' Specific message: Page Check Error in <i>object</i>	The database server detected inconsistencies with an index. Online log: Various messages depending on where the issue was detected. For example: Possible inconsistencies in a DBSpace TBLSpace Run 'oncheck -cD' on all DBSpace TBLSpaces Server state: Online. User action: Examine the online log file and run the recommended oncheck -cD command on the database.
Class ID: 2 Event ID: 2005	3	Class message: Index failure: ' <i>dbname:"owner".tabname:idxname</i> ' Specific message: An internal error occurred during batched index read because the database server had an invalid index key item.	
Class ID: 2 Event ID: 2006	3	Class message: Index failure: ' <i>dbname:"owner".tabname:idxname</i> ' Specific message: <i>index_page</i> log record too large to fit into the logical log buffer. Recommended minimum value for LOGBUFF is <i>number</i> .	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 2 Event ID: 2007	3	Class message: Index failure: <i>'dbname:"owner".tablename:idxname'</i> Specific message: Comparison based on locale <i>'locale_name'</i> failed	
Class ID: 2 Event ID: 2008	3	Class message: Index failure: <i>'dbname:"owner".tablename:idxname'</i> Specific message: Comparison failed	
Class ID: 2 Event ID: 2009	4	Class message: Index failure: <i>'dbname:"owner".tablename:idxname'</i> Specific message: An internal error occurred while the database server was trying to add a new item to the index.	The database server could not insert a record into an index. Online log: Assertion specifying the index and the recommended oncheck command to run. Server state: Online. User action: Examine the online log file and run the recommended oncheck command.
Class ID: 2 Event ID: 2010	4	Class message: Index failure: <i>'dbname:"owner".tablename:idxname'</i> Specific message: An internal error was raised due to an inconsistency in the index which is preventing the database server to position at the correct item in the index.	The database server could not retrieve the correct item in the index because of an inconsistency in the index. Online log: Assertion specifying the index and the recommended oncheck command to run. Server state: Online. User action: Examine the online log file and run the recommended oncheck command.
Class ID: 2 Event ID: 2011	3	Class message: Index failure: <i>'dbname:"owner".tablename:idxname'</i> Specific message: Cannot drop index <i>index_name</i> for partnum <i>partition_number</i> , iserrno = <i>error_number</i>	
Class ID: 2 Event ID: 2012	3	Class message: Index failure: <i>'dbname:"owner".tablename:idxname'</i> Specific message: An internal error occurred while the database server was trying to mark an index key descriptor as bad.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 2 Event ID: 2013	4	Class message: Index failure: <i>'dbname:"owner".tabname:idxname'</i> Specific message: An internal error occurred while the database server was trying to delete an item from the index.	The database server could not delete a record from an index. Online log: Assertion specifying the index and the recommended oncheck command to run. Server state: Online. User action: Examine the online log file and run the recommended oncheck command.
Class ID: 3 Event ID: 3001	3	Class message: Blob failure: <i>'dbname:"owner".tabname'</i> Specific message: tb_sockid in blob descriptor is corrupted. Current table is <i>'dbname:"owner".tabname'</i>	
Class ID: 3 Event ID: 3002	3	Class message: Blob failure: <i>'dbname:"owner".tabname'</i> Specific message: Incorrect BLOB stamps.	
Class ID: 3 Event ID: 3003	4	Class message: Blob failure: <i>'dbname:"owner".tabname'</i> Specific message: BLOB Page Check error at <i>dbspace_name</i>	The database server performed a check on pages that are moving between disk and memory and the check failed. Online log: Assertion describing the error. Server state: Online. User action: Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 3 Event ID: 3004	3	Class message: Blob failure: <i>'dbname:"owner".tabname'</i> Specific message: An internal error occurred while trying to read a blob from a table.	
Class ID: 3 Event ID: 3005	3	Class message: Blob failure: <i>'dbname:"owner".tabname'</i> Specific message: An internal error occurred while trying to copy a blob from a table.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 4 Event ID: 4001	4	Class message: Chunk is offline, mirror is active: <i>chunk_number</i> Specific message: I/O error, <i>error_number</i> Chunk ' <i>chunk_number</i> ' -- Offline	An error has occurred reading from or writing to a chunk. The database server has taken the chunk offline and switched to performing all I/O operations on the active mirrored chunk. Online log: Assertion describing the error that occurred. Server state: Online. User action: Examine the online log for information and fix the error. Run the onspaces -s command to recover the offline chunk. Retry the original operation. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 4 Event ID: 4002	3 or 4	Class message: Chunk is offline, mirror is active: <i>chunk_number</i> Specific message: An internal error occurred during physical I/O because the chunk was not opened.	The database server cannot access a chunk and switched to performing all I/O operations on the active mirrored chunk. Online log: Assertion describing the error and information about the chunk where the problem occurred. Server state: Online. User action: Examine the online log, fix any errors, and recover the mirror by using the onspaces utility. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 4 Event ID: 4003	3	Class message: Chunk is offline, mirror is active: <i>chunk_number</i> Specific message: I/O error, <i>error_number</i> Chunk ' <i>chunk_number</i> ' -- Offline (sanity)	
Class ID: 4 Event ID: 4004	3	Class message: Chunk is offline, mirror is active: <i>chunk_number</i> Specific message: Chunk failed sanity check	
Class ID: 4 Event ID: 4005	3	Class message: Chunk is offline, mirror is active: <i>chunk_number</i> Specific message: Mirror Chunk <i>chunk_number</i> added to space ' <i>space_number</i> '. Perform manual recovery.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 5 Event ID: 5001	4	Class message: Dbspace is offline: ' <i>dbspace_name</i> ' Specific message: Chunk <i>chunk_number</i> is being taken OFFLINE.	The database server took a dbspace offline because of an error in accessing a chunk. Online log: Assertion failure if the dbspace was a critical dbspace, such as the rootdbs. Assertion warning if the dbspace is not critical. Both provide information about the chunk and dbspace being taken offline. Server state: Online if a non-critical media failure. Offline if a critical media failure. User action: Examine the online log file and fix the underlying problem that caused the dbspace to be taken offline. You might need to restore the dbspace.
Class ID: 5 Event ID: 5002	4	Class message: Dbspace is offline: ' <i>dbspace_name</i> ' Specific message: WARNING! Chunk <i>chunk_number</i> is being taken OFFLINE for testing.	The database server has taken a dbspace offline as a result of an onmode command. Online log: Assertion warning indicating that the dbspace has been taken offline. Server state: Online. User action: None.
Class ID: 6 Event ID: 6016	3	Class message: Internal subsystem failure: ' <i>message</i> ' Specific message: Pool not freed. pool name: <i>pool_name</i> , address: <i>address</i>	
Class ID: 6 Event ID: 6017	4	Class message: Internal subsystem failure: ' <i>message</i> ' Specific message: CDR Grouper FanOut thread is aborting	A problem occurred with Enterprise Replication. Online log: Assertion describing the problem. Server state: Online. User action: Follow the instructions in the online log.
Class ID: 6 Event ID: 6018	4	Class message: Internal subsystem failure: ' <i>message</i> ' Specific message: CDR Pager: Paging File full: Waiting for additional space in CDR_QDATA_SBSPACE	The storage space of an Enterprise Replication queue is full. Online log: Assertion describing the problem. Server state: Online. User action: Add a chunk to one or more of the sbspaces specified by the CDR_QDATA_SBSPACE configuration parameter.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6021	3	Class message: Internal subsystem failure: 'message' Specific message: An internal error was reported by the database server during conversion when it found some indices in the old format.	
Class ID: 6 Event ID: 6022	3	Class message: Internal subsystem failure: 'message' Specific message: An internal error was reported by the database server when it checks for any new in-place alter pending in the current server during reversion.	
Class ID: 6 Event ID: 6023	3	Class message: Internal subsystem failure: 'message' Specific message: Cannot open index 'dbname:index_name', iserrno = error_number	
Class ID: 6 Event ID: 6024	3	Class message: Internal subsystem failure: 'message' Specific message: Cannot drop index 'dbname:index_name', iserrno = error_number	
Class ID: 6 Event ID: 6025	3	Class message: Internal subsystem failure: 'message' Specific message: Cannot open table 'dbname:table_name', iserrno = error_number	
Class ID: 6 Event ID: 6026	3	Class message: Internal subsystem failure: 'message' Specific message: Cannot drop table 'dbname:table_name', iserrno = error_number	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6027	3	Class message: Internal subsystem failure: 'message' Specific message: An error was reported by the database server when it tried to drop the sysmaster database during reversion	
Class ID: 6 Event ID: 6028	3	Class message: Internal subsystem failure: 'message' Specific message: Optical Subsystem STARTUP Error	
Class ID: 6 Event ID: 6029	2	Class message: Internal subsystem failure: 'message' Specific message: Unable to initiate communications with the Optical Subsystem	
Class ID: 6 Event ID: 6030	3	Class message: Internal subsystem failure: 'message' Specific message: Invalid or missing name for Subsystem Staging BLOBspace	
Class ID: 6 Event ID: 6031	2	Class message: Internal subsystem failure: 'message' Specific message: Optical Subsystem is running	
Class ID: 6 Event ID: 6032	2	Class message: Internal subsystem failure: 'message' Specific message: Optical Subsystem is not running	
Class ID: 6 Event ID: 6033	5	Class message: Internal subsystem failure: 'message' Specific message: Cache read error	The database server shut down after encountering an error while reading an internal cache. Online log: Assertion Failure. Server State: Offline. User action: Start the database server and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6034	3	Class message: Internal subsystem failure: 'message' Specific message: Could not start remote server	
Class ID: 6 Event ID: 6035	3	Class message: Internal subsystem failure: 'message' Specific message: An error was reported by the database server during the handling of audit trail files.	
Class ID: 6 Event ID: 6036	3	Class message: Internal subsystem failure: 'message' Specific message: Archive on <i>dbspaces_list</i> ABORTED	
Class ID: 6 Event ID: 6037	3	Class message: Internal subsystem failure: 'message' Specific message: Waiting on BLOBSpace to appear for Logical Recovery	
Class ID: 6 Event ID: 6038	3	Class message: Internal subsystem failure: 'message' Specific message: An internal error reported by the database server. Users may need to look at the specific message which accompanies with this id.	
Class ID: 6 Event ID: 6039	3	Class message: Internal subsystem failure: 'message' Specific message: Wrong page for cleaning deleted items	
Class ID: 6 Event ID: 6040	3	Class message: Internal subsystem failure: 'message' Specific message: Buffer in wrong state for cleaning deleted items	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6041	5 or 3	Class message: Internal subsystem failure: 'message' Specific message: An internal error was detected by the Buffer Manager in the database server.	<p>For severity 5, the database server buffer manager encountered an internal error and either shut down or corrected the problem.</p> <p>Online log: Assertion warning or an assertion failure with a description of the operation being performed at the time of the error. Typically, an assertion warning shows that the error was internally corrected.</p> <p>Server State: Offline if the error was unrecoverable. Online if the error was corrected.</p> <p>User action: If the error was unrecoverable, start the database server and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support. No action is required if the error was internally corrected by the database server.</p>
Class ID: 6 Event ID: 6042	5 or 2	Class message: Internal subsystem failure: 'message' Specific message: An internal error was reported by the database server when it detected an inconsistency with the internal buffer queues.	<p>For severity 5, the database server detected an inconsistency during the processing of internal buffer queues and either shut down or corrected the problem.</p> <p>Online log: Assertion warning or an assertion failure with a description of the operation being performed at the time of the error. Typically, an assertion warning shows that the error was internally corrected.</p> <p>Server State: Offline if the error was unrecoverable. Online if the error was corrected.</p> <p>User action: If the error was unrecoverable, start the database server and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support. No action is required if the error was internally corrected by the database server.</p>
Class ID: 6 Event ID: 6043	3	Class message: Internal subsystem failure: 'message' Specific message: Internal filep error	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6044	3	Class message: Internal subsystem failure: 'message' Specific message: An internal error was corrected automatically by the database server when it tried to save the log buffer into a system log buffer.	
Class ID: 6 Event ID: 6045	5	Class message: Internal subsystem failure: 'message' Specific message: Logical logging error for 'object' in 'space'	The database server shut down because of an error while processing logical logs. Online log: Assertion failure with a description of the operation and logical log information. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6046	4	Class message: Internal subsystem failure: 'message' Specific message: Page Check Error in <i>object</i>	The database server detected inconsistencies in the data. Online log: Various outputs depending upon where the issue was detected. For example: Possible inconsistencies in a DBSpace TBLSpace Run 'oncheck -cD' on all DBSpace TBLSpaces Server state: Online. User action: Examine the online log file and run the recommended oncheck -cD command on the database.
Class ID: 6 Event ID: 6047	3	Class message: Internal subsystem failure: 'message' Specific message: Errors occurred while recreating indexes	
Class ID: 6 Event ID: 6049	5	Class message: Internal subsystem failure: 'message' Specific message: Lock types <i>lock_type</i> and <i>lock_type</i> should never be merged	The database server shut down after attempting to merge incompatible locks. Online log: Assertion failure with the lock types that the database server was attempting to merge. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 5 6 Event ID: 6050	5	Class message: Internal subsystem failure: 'message' Specific message: An internal error was reported by the database server when it detected some corruption in the lock free list chain.	The database server shut down after detecting corruption of an internal structure that manages an internal list of free locks. Online log: Assertion failure. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 3 6 Event ID: 6051	3	Class message: Internal subsystem failure: 'message' Specific message: ERROR - NO 'waitfor' locks in Critical Section!!!	
Class ID: 3 6 Event ID: 6052	3	Class message: Internal subsystem failure: 'message' Specific message: Internal Tblspace error	
Class ID: 3 6 Event ID: 6053	3	Class message: Internal subsystem failure: 'message' Specific message: Session does not have exclusive access to partition <i>partition_name</i> . Request to drop the partition ignored.	
Class ID: 3 6 Event ID: 6054	3	Class message: Internal subsystem failure: 'message' Specific message: Error building 'sysmaster' database.	
Class ID: 3 6 Event ID: 6055	3	Class message: Internal subsystem failure: 'message' Specific message: Setread error on SMI Table, partnum <i>partition_number</i>	
Class ID: 3 6 Event ID: 6056	3	Class message: Internal subsystem failure: 'message' Specific message: Comparison based on locale ' <i>locale_name</i> ' failed	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6057	2	Class message: Internal subsystem failure: 'message' Specific message: DBSPACETEMP internal list not initialized, using default	
Class ID: 6 Event ID: 6058	3	Class message: Internal subsystem failure: 'message' Specific message: A data source accessed using a gateway (<i>gateway_name</i>) might be in an inconsistent state	
Class ID: 6 Event ID: 6059	3	Class message: Internal subsystem failure: 'message' Specific message: Prepared participant site <i>site_name</i> not responding	
Class ID: 6 Event ID: 6060	5	Class message: Internal subsystem failure: 'message' Specific message: Thread exited with <i>number</i> buffers held	The database server shut down after detecting that a thread is holding one or more buffers. Online log: Assertion failure with the number of buffers being held by the thread. Server State: Offline. User action: Bring the database server online. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6061	3	Class message: Internal subsystem failure: 'message' Specific message: An internal error was automatically corrected by the database server when it detected that the undo log for the transaction was not applicable.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 5 6 Event ID: 6062	5	Class message: Internal subsystem failure: 'message' Specific message: Internal Error - Freeing transaction entry that still holds locks!	While freeing resources associated with a transaction, the database server detected that the transaction is holding locks. In some circumstances the database server can release these locks. Online log: Assertion failure with the transaction or an assertion warning with the transaction and a statement that the database server internally corrected the problem. Server State: Offline if the error was unrecoverable. Online if the error was internally corrected. User action: If the database server shut down, start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6063	3	Class message: Internal subsystem failure: 'message' Specific message: User thread not on TX wait list	
Class ID: 6 Event ID: 6064	3	Class message: Internal subsystem failure: 'message' Specific message: Due to a heuristic decision, the work done on behalf of the specified transaction branch might have been heuristically completed or committed or rolled back or partially committed and partially rolled back.	
Class ID: 6 Event ID: 6065	3	Class message: Internal subsystem failure: 'message' Specific message: Errors occurred while recreating indexes	
Class ID: 6 Event ID: 6066	3	Class message: Internal subsystem failure: 'message' Specific message: An internal error is reported by the database server when it has checked all sites to see if a heuristic rollback was the reason for the failure.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6067	5	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (Recursive exception) has caused the database server processes to terminate unexpectedly.	The database server detected recursive calls to exception handling and immediately shut down to avoid an infinite loop. Online log: Assertion failure. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6068	5	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (Internal exception) has caused the database server processes to terminate unexpectedly.	The database server shut down due to an unrecoverable internal error. Online log: Assertion failure with information about the exception that caused the problem. Server State: Offline. User action: Start the database server. Look at the exception information in the assertion failure file. If the exception relates to a user-defined routine, investigate and correct the user-defined routine. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6069	5	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (Master daemon died) has caused the database server processes to terminate unexpectedly.	The master daemon oninit process stopped and the database server shut down. This error can be caused by the termination of operating system processes. Online log: Assertion failure. Server State: Offline. User action: Start the database server. Be careful when terminating operating system processes.
Class ID: 6 Event ID: 6070	5	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (VP died) has caused the database server processes to terminate unexpectedly.	An oninit process stopped and the database server shut down. This error can be caused by the termination of operating system processes. Online log: Assertion failure. Server State: Offline. User action: Start the database server. Be careful when terminating operating system processes.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 5 6 Event ID: 6071	5	Class message: Internal subsystem failure: 'message' Specific message: ERROR: can not fork secondary Server thread (MACH11 Shutdown)	The secondary server shut down but was unable to create a thread to shut down normally. Online log: DR: Shutting down the server. ERROR: can not fork secondary Server thread (MACH11 Shutdown) Can not run onmode -ky PANIC: Attempting to bring system down. Server State: Offline. User action: None.
Class ID: 6 Event ID: 6072	3	Class message: Internal subsystem failure: 'message' Specific message: Generic unique event id when the server failed to fork a new thread.	
Class ID: 6 Event ID: 6073	3	Class message: Internal subsystem failure: 'message' Specific message: An error was reported by the database server when it could not initialize GLS for starting a session.	
Class ID: 6 Event ID: 6074	3	Class message: Internal subsystem failure: 'message' Specific message: WARNING: mt_aio_wait: errno == EINVAL	
Class ID: 6 Event ID: 6075	5	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (KAIO) has caused the database server processes to terminate unexpectedly.	The database server shut down because of an error in the KAIO subsystem. Online log: Assertion failure with the specific operation that failed. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6100		Generic event for when the database server implicitly raises an assert warning.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 6 Event ID: 6300		Generic event for when the database server implicitly raises an assert failure.	
Class ID: 6 Event ID: 6500		Generic event for when the database server terminates unexpectedly due to an internal error condition.	<p>An internal error occurred and the database server shut down.</p> <p>Online log: Assertion failure.</p> <p>Server State: Offline.</p> <p>User action: Start the database server. Examine the assertion failure file for more information about what happened. If possible, fix any problems identified and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.</p>
Class ID: 7 Event ID: 7001	3	<p>Class message:</p> <p>Database server initialization failure</p> <p>Specific message:</p> <p>TABLOCKS log record too large to fit into the logical log buffer. Recommended minimum value for LOGBUFF is <i>size</i>.</p> <p>I-STAR(C) begins prepare log record too large to fit into the logical log buffer. Recommended minimum value for LOGBUFF is <i>size</i>.</p> <p>Partition blob log record too large to fit into the logical log buffer. Recommended minimum value for LOGBUFF is <i>size</i>.</p> <p>Alter table special column desc log record too large to fit into the logical log buffer. Recommended minimum value for LOGBUFF is <i>size</i>.</p>	
Class ID: 7 Event ID: 7002	4	<p>Class message:</p> <p>Database server initialization failure</p> <p>Specific message:</p> <p>Unable to extend <i>number</i> reserved pages for checkpoint in ROOT chunk.</p> <p>Unable to extend <i>number</i> reserved pages for log in ROOT chunk.</p>	<p>The database server could not start because it could not allocate more space for internal structures in the initial root chunk.</p> <p>Online log: Assertion.</p> <p>Server state: Offline.</p> <p>User action: Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.</p>

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 7 Event ID: 7003	4	Class message: Database server initialization failure Specific message: An internal error occurred during conversion. Users may need to take a look at the specific messages for further action.	The database server could not start during an upgrade because an internal error occurred during the conversion process. Online log: Assertion describing the error. Server state: Offline. User action: Look at the online log and the specific message and take the necessary corrective action. Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 7 Event ID: 7004	4	Class message: Database server initialization failure Specific message: An internal error occurred while trying to convert the database tblspace.	The database server cannot start because of an internal error when trying to convert the database tblspace, which holds information about the databases in the instance. Online log: Assertion describing the error. Server state: Offline. User action: Contact IBM Software Support.
Class ID: 7 Event ID: 7005	4	Class message: Database server initialization failure Specific message: An internal error occurred while trying to convert blob free map pages.	The database server cannot start because of an internal error when trying to convert blob space free-map pages. Online log: Assertion describing the error. Server state: Offline. User action: Contact IBM Software Support.
Class ID: 7 Event ID: 7006	4	Class message: Database server initialization failure Specific message: Cannot Open Logical Log.	The database server cannot start because it is still restoring physical or logical logs. This situation can occur if the onmode -m or onmode -s command is run before the restore is complete. Online log: Assertion describing the error. Server state: Recovering and starting. User action: Run the onmode -m or onmode -s command after the restore is complete.
Class ID: 7 Event ID: 7007	4	Class message: Database server initialization failure Specific message: Logical Log File not found.	The database server cannot start because a logical log file is missing. Online log: Assertion describing the error. Server state: Offline. User action: Restore the database server from a backup.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 7 Event ID: 7008	3	Class message: Database server initialization failure Specific message: WARNING! LTXHWM is set to 100%. This long transaction high water mark will never be reached. Transactions will not be aborted automatically by the server, regardless of their length.	
Class ID: 7 Event ID: 7009	4	Class message: Database server initialization failure Specific message: A Physical or Logical Restore is active.	The database server cannot start because it is still restoring physical or logical logs. This situation can occur if the onmode -m or onmode -s command is run before the restore is complete. Online log: Assertion describing the error. Server state: Recovering and starting. User action: Run the onmode -m or onmode -s command after the restore is complete.
Class ID: 7 Event ID: 7010	4	Class message: Database server initialization failure Specific message: <i>root_dbspace</i> has not been physically recovered.	The database server cannot start because the restore was interrupted before the rootdbs was physically restored. Online log: Assertion describing the error. Server state: Offline. User action: Restore the rootdbs.
Class ID: 7 Event ID: 7011	4	Class message: Database server initialization failure Specific message: <i>dbspace</i> has not been physically recovered.	The database cannot start because a dbspace is not physically restored. This situation can occur if the database server is attempted to be started before a restore is complete. Online log: Assertion describing the error. Server state: Offline. User action: Wait until the restore is complete before starting the database server.
Class ID: 7 Event ID: 7012	4	Class message: Database server initialization failure Specific message: <i>dbspace</i> not recovered from same archive backup as <i>dbspace</i> .	The database server cannot start because a dbspace was not restored successfully. Online log: Assertion describing the error. Server state: Offline. User action: Recover the dbspace from a backup and roll forward the necessary logs to bring the dbspace to the correct point in time.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 7 Event ID: 7013	4	Class message: Database server initialization failure Specific message: Log <i>log_number</i> not found.	The database server cannot start because a restore is not complete. Online log: Assertion describing the error. Server state: Offline. User action: Wait until the restore is complete before starting the database server.
Class ID: 7 Event ID: 7014	4	Class message: Database server initialization failure Specific message: Logical restore cannot be skipped. Perform a logical restore.	The database server cannot start because a logical restore is not complete. Online log: Assertion describing the error. Server state: Offline. User action: Perform a logical restore (for example, by using the onbar -r -l command) and start the database server in quiescent or online mode.
Class ID: 7 Event ID: 7015	4	Class message: Database server initialization failure Specific message: Cannot change to On-Line or Quiescent mode.	The database server cannot start because of an error during fast or full recovery. Online log: Assertion describing the error. Server state: Offline. User action: Examine the online log for more information. Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 7 Event ID: 7016	4	Class message: Database server initialization failure Specific message: Cannot Open Primary Chunk ' <i>chunk_number</i> '.	The database server cannot start because it could not access a primary chunk. Online log: Assertion describing the error. Server state: Offline. User action: Examine the online log for more information. Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 7 Event ID: 7017	4	Class message: Database server initialization failure Specific message: The chunk ' <i>chunk_number</i> ' must have owner-ID " <i>owner_id</i> " and group-ID " <i>group_id</i> ".	The database server cannot start because the owner and group of a chunk path are not correct. Online log: Assertion describing the problem. Server state: Offline. User action: Correct the permissions on the chunk path mentioned in the specific message. Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 7 Event ID: 7018	4	Class message: Database server initialization failure Specific message: The chunk ' <i>chunk_number</i> ' must have READ/WRITE permissions for owner and group (660).	The database server cannot start because the permissions on a chunk path are not correct. Online log: Assertion describing the problem. Server state: Offline. User action: Correct the permissions on the chunk path mentioned in the specific message. Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 7 Event ID: 7019	4	Class message: Database server initialization failure Specific message: Memory allocation error.	The database server cannot start because it failed to allocate enough memory. Online log: Server state: Offline. User action: Ensure that enough memory is available for the configuration you have specified for the database server. Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 7 Event ID: 7020	4	Class message: Database server initialization failure Specific message: The chunk ' <i>chunk_number</i> ' will not fit in the space specified.	The database server cannot start because there is insufficient space to create the specified chunk. Online log: Assertion describing the problem. Server state: Offline. User action: Specify a smaller size for the chunk or free additional space for the chunk.
Class ID: 7 Event ID: 7021	4	Class message: Database server initialization failure Specific message: <i>device_name</i> : write failed, file system is full.	The database server cannot start because the file system does not have free space. Online log: Assertion describing the problem. Server state: Offline. User action: Ensure the file system mentioned in the specific message has enough space. Retry the original operation. If the operation fails again, note all circumstances and contact IBM Software Support.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 7 Event ID: 7022	3	Class message: Database server initialization failure Specific message: An error occurred while the database server was creating the SMI database.	
Class ID: 7 Event ID: 7023	4	Class message: Database server initialization failure Specific message: Unable to create boot strap config file - 'file_name'	The database server cannot start because it could not create a configuration file. Online log: Assertion describing the error. Server state: Offline. User action: Examine the online log for more information and fix the problem. The problem might be incorrect permissions on a directory. Retry the original operation. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 7 Event ID: 7024	3	Class message: Database server initialization failure Specific message: 'sysmaster' database will not be built/checked	
Class ID: 7 Event ID: 7025	3	Class message: Database server initialization failure Specific message: WARNING! Physical Log size <i>size</i> is too small. Physical Log overflows may occur during peak activity. Recommended minimum Physical Log size is <i>number</i> times maximum concurrent user threads.	
Class ID: 7 Event ID: 7026	3	Class message: Database server initialization failure Specific message: WARNING! Logical log layout may cause __ISN__ to get into a locked state. Recommended smallest logical log size is <i>number</i> times maximum concurrent user threads.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 7 Event ID: 7027	3	Class message: Database server initialization failure Specific message: WARNING! Buffer pool size may cause __ISN__ to get into a locked state. Recommended minimum buffer pool size is <i>number</i> times maximum concurrent user threads.	
Class ID: 7 Event ID: 7028	3	Class message: Database server initialization failure Specific message: Checkpoint log record may not fit into the logical log buffer. Recommended minimum value for LOGBUFF is <i>size</i> .	
Class ID: 7 Event ID: 7029	3	Class message: Database server initialization failure Specific message: Temp transaction not NULL.	
Class ID: 9 Event ID: 9001	4	Class message: Physical recovery failure Specific message: Physical log recovery error	The physical recovery of the database server failed. Online log: Assertion failure with a description of the problem. Server state: Online. User action: Retry the operation or restore from a backup.
Class ID: 10 Event ID: 10001	3 or 4	Class message: Logical recovery failure Specific message: Rollback error <i>error_number</i>	Logical recovery failed because the database server could not roll back a transaction. Online log: Assertion with details of the error and the log or log record where the problem occurred. Server state: Online or offline, depending on the error. User action: Examine the online log file for more information and run any recommended commands, such as an oncheck command. Retry the original operation. If the operation fails again, note all circumstances and contact IBM Software Support.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 10 Event ID: 10002	4	Class message: Logical recovery failure Specific message: Logical Recovery ABORTED.	The logical recovery of the database server failed. Online log: Assertion warning with information about the log record. Assertion failure with information about the log record if the failure is associated with a critical dbspace. Server state: Online if the dbspace is not critical. Offline if the dbspace is critical. User action: Examine the online log to determine the appropriate action, for example, you might need to restart the warm restore.
Class ID: 10 Event ID: 10003	4	Class message: Logical recovery failure Specific message: Log record (<i>log_subsystem:log_type</i>) in log <i>log_number</i> , offset <i>log_position</i> was not rolled back	Logical recovery encountered an internal error while rolling back a transaction. Online log: Message describing the log record. Server state: Online. User action: Examine the online log and determine the appropriate action, for example, resubmit the transaction.
Class ID: 10 Event ID: 10004	3	Class message: Logical recovery failure Specific message: Logical Logging error for ' <i>log_subsystem:log_type</i> ' in ' <i>object</i> '	
Class ID: 10 Event ID: 10005	4	Class message: Logical recovery failure Specific message: An internal error occurred while trying to apply the log records during logical log recovery.	Logical recovery failed. Online log: Assertion warning with information about the log record. Server state: Depends on the failure. User action: Examine the online log and determine the appropriate action, for example, restart the warm restore.
Class ID: 10 Event ID: 10006	3 or 4	Class message: Logical recovery failure Specific message: An internal error occurred when the database server tried to find the file descriptor for the tblspace.	Logical recovery failed because the database server could not find an internal file descriptor for a partition. Online log: Assertion indicating for which table the error occurred and instructions to run the oncheck command. Server state: Online. User action: Run the oncheck -cDI command for the table mentioned in the online log or for the database.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 11	3	Class message: Cannot open chunk: ' <i>pathname</i> '	
Event ID: 11001		Specific message: Cannot Open Mirror Chunk ' <i>chunk_number</i> ', <i>errno</i> = <i>error_number</i>	
Class ID: 11	3	Class message: Cannot open chunk: ' <i>pathname</i> '	
Event ID: 11002		Specific message: Cannot Open Primary Chunk ' <i>chunk_number</i> ', <i>errno</i> = <i>error_number</i>	
Class ID: 12	3	Class message: Cannot open dbspace: ' <i>dbspace_name</i> '	
Event ID: 12001		Specific message: ERROR: Dbspace <i>dbspace_name</i> not found among table <i>table_name</i> fragments.	
Class ID: 13	2	Class message: Performance improvement possible	
Event ID: 13001		Specific message: The number of configured CPU poll threads exceeds number of CPU VPs specified in 'VPCLASS <i>cpu</i> '. NETTYPE ' <i>protocol</i> ' poll threads started on NET VPs.	
Class ID: 13	2	Class message: Performance improvement possible	
Event ID: 13002		Specific message: Transaction table overflow due to parallel recovery.	
Class ID: 14	3	Class message: Database failure. ' <i>dbname</i> '	
Event ID: 14001		Specific message: " <i>dbname</i> " - Error <i>error_number</i> during logging mode change.	
Class ID: 15	3	Class message: High-Availability Data-Replication failure	
Event ID: 15001		Specific message: DR: Turned off on secondary server	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 15 Event ID: 15002	3	Class message: High-Availability Data-Replication failure Specific message: DR: Turned off on primary server	
Class ID: 15 Event ID: 15003	3	Class message: High-Availability Data-Replication failure Specific message: DR: Cannot connect to secondary server	
Class ID: 15 Event ID: 15004	3	Class message: High-Availability Data-Replication failure Specific message: DR: Received connection request from remote server when DR is not Off [Local type: <i>type</i> , Current state: <i>state</i>] [Remote type: <i>type</i>]	
Class ID: 15 Event ID: 15005	3	Class message: High-Availability Data-Replication failure Specific message: DR: Received connection request before physical recovery completed.	
Class ID: 15 Event ID: 15006	3	Class message: High-Availability Data-Replication failure Specific message: DR: Local and Remote server type and/or last change (LC) incompatible [Local type: <i>type</i> , LC: <i>type</i>] [Remote type: <i>type</i> , LC: <i>type</i>]	
Class ID: 16 Event ID: 16001	2	Class message: Backup completed: ' <i>dbspace_list</i> ' Specific message: Archive on <i>dbspace_list</i> completed without being recorded.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 16	2	Class message: Backup completed: 'dbspace_list'	
Event ID: 16002		Specific message: Archive on <i>dbspace_list</i> Completed with <i>number</i> corrupted pages detected.	
Class ID: 16	2	Class message: Backup completed: 'dbspace_list'	
Event ID: 16003		Specific message: Archive on <i>dbspace_list</i> Completed	
Class ID: 17	4	Class message: Backup aborted: 'dbspace_list'	The database server detected corruption and stopped the backup.
Event ID: 17001		Specific message: Archive detects that page <i>chunk_number:page_offset</i> is corrupt.	Online log: Assertion describing the problem. Server state: Online. User action: Examine the online log for information about the corruption. Try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 17	3	Class message: Backup aborted: 'dbspace_list'	
Event ID: 17002		Specific message: Page %d:%d of partition <i>partition_number</i> not archived.	
Class ID: 18	2	Class message: Log backup completed: <i>log_number</i>	
Event ID: 18001		Specific message: Logical Log <i>log_number</i> - Backup Completed	
Class ID: 19	3	Class message: Log backup aborted: <i>log_number</i>	
Event ID: 19001		Specific message: Logical Log <i>log_number</i> - Backup Aborted <i>message</i>	
Class ID: 20	3	Class message: Logical logs are full—backup is needed	
Event ID: 20001		Specific message: Logical Log Files are Full -- Backup is Needed	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 20 Event ID: 20002	3	Class message: Logical logs are full—backup is needed Specific message: Waiting for Next Logical Log File to be Freed	
Class ID: 20 Event ID: 20003	3	Class message: Logical logs are full—backup is needed Specific message: Logical Log Files are almost Full -- Backup is Needed. In Data replication scenario, this could block failure-recovery of the paired server.	
Class ID: 21 Event ID: 21001	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Archive arcbu_next_tbuf() – Buffer Overflow	
Class ID: 21 Event ID: 21002	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Archive tcp_logbu_hdr() – Buffer Overflow	
Class ID: 21 Event ID: 21003	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Archive tcp_logbu_trl() – Buffer Overflow	
Class ID: 21 Event ID: 21004	2 or 5	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Physical log file overflow	For severity 5, the physical log file is full and needs to overflow. If this happens during recovery, the database server attempts to extend the physical log. Online log: Assertion failure if the database server is either not in recovery or is unable to extend the physical log. Assertion warning if the database server is in recovery and extends the physical log. Server State: Offline. User action: None.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 21 Event ID: 21005	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Lock table overflow - user id %d, session id %d	
Class ID: 21 Event ID: 21006	5	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Logical log buffer overflow detected	The database server shut down because the logical log buffer is full. Online log: Assertion failure with the log record size and the buffer size. Server State: Offline. User action: Increase the value of the LOGBUFF configuration parameter in the onconfig file. Start the database server.
Class ID: 21 Event ID: 21007	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Llog logbu_logfile() – Buffer Overflow	
Class ID: 21 Event ID: 21008	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Llog logbu_bpage() – Buffer Overflow	
Class ID: 21 Event ID: 21009	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Unable to allocate a user thread for user id <i>user_ID</i>	
Class ID: 21 Event ID: 21010	3	Class message: Database server resource overflow: <i>'resource_name'</i> Specific message: Unable to allocate a transaction for user id <i>user_ID</i> , session id <i>session_ID</i>	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 22	3	Class message: Long transaction detected	
Event ID: 22001		Specific message: Blocking on XA transaction, tx <i>transaction_number</i> , till it is cleaned up.	
Class ID: 22	3	Class message: Long transaction detected	
Event ID: 22002		Specific message: Continuing Long Transaction (for COMMIT): tx:	
Class ID: 22	3	Class message: Long transaction detected	
Event ID: 22003		Specific message: Aborting Long Transaction: tx:	
Class ID: 23	2	Class message: Logical log ' <i>number</i> ' complete	
Event ID: 23001		Specific message: Logical Log <i>log_number</i> Complete, timestamp: <i>timestamp</i> .	
Class ID: 24	3	Class message: Unable to allocate memory	
Event ID: 24001		Specific message: Generic unique event id when the server failed to allocate memory for starting a new thread.	
Class ID: 24	3	Class message: Unable to allocate memory	
Event ID: 24002		Specific message: Warning: unable to allocate requested big buffer of size <i>size</i>	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 24 Event ID: 24003	3	Class message: Unable to allocate memory Specific message: The database server tried to allocate a shared memory virtual segment before it was actually needed, in accordance with the setting of the SHMVIRT_ALLOCSEG configuration parameter - but the segment could not be added. Next failure message will be printed in 30 minutes.	
Class ID: 24 Event ID: 24004	3	Class message: Unable to allocate memory Specific message: out of message shared memory	
Class ID: 24 Event ID: 24005	3	Class message: Unable to allocate memory Specific message: out of message shared memory	
Class ID: 24 Event ID: 24006	3	Class message: Unable to allocate memory Specific message: out of virtual shared memory	
Class ID: 24 Event ID: 24007	3	Class message: Unable to allocate memory Specific message: No memory available for page cleaners	
Class ID: 24 Event ID: 24008	3	Class message: Unable to allocate memory Specific message: kysearch(): Memory allocation error	
Class ID: 24 Event ID: 24009	3	Class message: Unable to allocate memory Specific message: Lock table overflow - user id <i>user_ID</i> , session id <i>session_ID</i>	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 24	3	Class message: Unable to allocate memory	
Event ID: 24010		Specific message: Unable to allocate a user thread for user id <i>user_ID</i>	
Class ID: 24	3	Class message: Unable to allocate memory	
Event ID: 24011		Specific message: Unable to allocate a transaction for user id <i>user_ID</i> , session id <i>session_ID</i>	
Class ID: 25	2	Class message: Internal subsystem initialized: ' <i>message</i> ' (starts the optical subsystem)	
Event ID: 25001		Specific message: Optical Subsystem is running.	
Class ID: 26	3	Class message: Dynamically added log file <i>logid</i>	
Event ID: 26001		Specific message: Dynamically added log file <i>logid</i> to DBspace <i>dbspace_name</i>	
Class ID: 27	4	Class message: Log file required	The database server needs an additional log file to continue processing.
Event ID: 27001		Specific message: ALERT: The oldest logical log (<i>log_number</i>) contains records from an open transaction (<i>transaction_number</i>). Logical logging will remain blocked until a log file is added. Add the log file with the onparams -a command, using the -i (insert) option, as in: onparams -a -d <i>dbspace</i> -s <i>size</i> -i Then complete the transaction as soon as possible.	
			Online log: ALERT: The oldest logical log (<i>log_number</i>) contains records from an open transaction (<i>transaction_number</i>). Logical logging will remain blocked until a log file is added. Add the log file with the onparams -a command, using the -i (insert) option, as in: onparams -a -d <i>dbspace</i> -s <i>size</i> -i Then complete the transaction as soon as possible. Server state: Online. User action: Add a new logical log.

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 28 Event ID: 28001	4	Class message: No space for log file Specific message: ALERT: Because the oldest logical log (<i>log_number</i>) contains records from an open transaction (<i>transaction_number</i>), the server is attempting to dynamically add a log file. But there is no space available. Please add a DBspace or chunk. Then complete the transaction as soon as possible.	The database server cannot dynamically add an additional logical log file because not enough space is available. Online log: Assertion warning indicating that there is not enough space available for an additional logical log file. Server state: Online. User action: Add a new logical log file or additional space.
Class ID: 28 Event ID: 28002	4	Class message: No space for log file Specific message: Warning - Enterprise Replication is attempting to dynamically add a log file. But there is no space available. The replay position may overrun.	The database server cannot dynamically add an additional logical log file because not enough space is available. Online log: Assertion warning indicating that there is not enough space available for an additional logical log file. Server state: Online. User action: Add a new logical log file or additional space.
Class ID: 29 Event ID: 29001	2	Class message: Internal subsystem: <i>subsystem</i> Specific message: Skipped existing audit trail files <i>file_name</i> to <i>file_name</i> .	
Class ID: 30 - 39	2, 3, or 4	Enterprise Replication events. See Enterprise Replication event alarms.	
Class ID: 40 Event ID: 40001	3	Class message: RSS alarm Specific message: RSS <i>server_name</i> added	
Class ID: 40 Event ID: 40002	3	Class message: RSS alarm Specific message: Password for RSS Source <i>server_name</i> changed	
Class ID: 40 Event ID: 40003	3	Class message: RSS alarm Specific message: RSS <i>server_name</i> deleted	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 40	3	Class message: RSS alarm	
Event ID: 40004		Specific message: RSS <i>server_name</i> log replay position is falling too far behind RSS Source	
Class ID: 40	3	Class message: RSS alarm	
Event ID: 40005		Specific message: RSS <i>server_name</i> is not acknowledging log transmission	
Class ID: 40	3	Class message: RSS alarm	
Event ID: 40006		Specific message: Error receiving a buffer from RSS <i>server_name</i> - shutting down	
Class ID: 41	3	Class message: SDS alarm	
Event ID: 41001		Specific message: ERROR: Removing SDS Node <i>server_name</i> has timed out - removing	
Class ID: 42	1	Class message: Event occurred	
Event ID: 42001			
Class ID: 43	3	Class message: Connection Manager alarm	
Event ID: 43001		Specific message: CM:Session for Connection manager <i>name</i> terminated abnormally	
Class ID: 44	3	Class message: DBSpace is full: <i>dbspace_name</i>	
Event ID: 44001		Specific message: WARNING: <i>dbspace_type dbspace_name</i> is full	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 45	3	Class message: partition ' <i>partition_name</i> ': no more extents	
Event ID: 45001		Specific message: Partition ' <i>partition_name</i> ': No more extents	
Class ID: 46	3	Class message: partition ' <i>partition_name</i> ': no more pages	
Event ID: 46001		Specific message: Partition ' <i>partition_name</i> ': No more pages	
Class ID: 47 - 71	3 or 4	Enterprise Replication events. See Enterprise Replication event alarms.	
Class ID: 72	2	Class message: Audit trail is switched to a new file.	
Event ID: 72001		Specific message: Audit trail switched to <i>file_name</i>	
Class ID: 73-77	3 or 4	Enterprise Replication events. See Enterprise Replication event alarms.	
Class ID: 78	3	Class message: The storage pool is empty.	
Event ID: 78001		Specific message: Warning: The storage pool is out of space.	
Class ID: 79	3	Class message: Dynamically added chunk <i>chunk_name</i> to space	
Event ID: 79001		Specific message: Dynamically added chunk <i>chunk_name</i> to space ' <i>space_name</i> ' Path: <i>path</i> , offset <i>offset_number</i> kilobytes Size: <i>size</i> kilobytes	
Class ID: 80		Not currently in use.	

Table C-3. Event Alarms (continued)

ID	Severity	Messages	Explanation
Class ID: 81 Event ID: 81001	4	Class message: <p>Logical log file or dbspace corruption detected during backup. Loguniq or Dbspace id: <i>ID</i>.</p> Specific message: <p>Log Backup detected a corrupted logical log file.</p> <p>Expected loguniq:pagenum <i>log_number:page_number</i></p> <p>Actual loguniq:pagenum <i>log_number:page_number</i></p> <p>Log backup continuing but the log backup cannot be used to restore a server.</p> <p>You should run oncheck and take a level 0 archive.</p>	<p>A backup failed because the database server detected corruption in the logical log file or dbspace.</p> <p>Online log: Assertion warning.</p> <p>Server state: Online.</p> <p>User action: Perform a new level-0 backup.</p>
Class ID: 82 Event ID: 82001	3	Class message: <p>session <i>ID (thread)</i> network write operation has been blocked for at least 30 minutes, which might indicate an operating system problem</p> Specific message: <p>session <i>ID (thread)</i> network write operation has been blocked for at least 30 minutes, which might indicate an operating system problem</p>	
Class ID: 83 Event ID: 83001	3	Class message: <p>SDS: Failover aborted - detected primary server is still active.</p> Specific message: <p>SDS: Failover aborted - detected primary server is still active.</p>	

Severity 5 event alarms

Severity 5 event alarms indicate that the database server has failed.

Table C-4. Severity 5 event alarms

ID	Messages	Explanation
Class ID: 6 Event ID: 6033	Class message: Internal subsystem failure: 'message' Specific message: Cache read error	<p>The database server shut down after encountering an error while reading an internal cache.</p> <p>Online log: Assertion Failure.</p> <p>Server State: Offline.</p> <p>User action: Start the database server and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.</p>
Class ID: 6 Event ID: 6041	Class message: Internal subsystem failure: 'message' Specific message: An internal error was detected by the Buffer Manager in the database server.	<p>The database server buffer manager encountered an internal error and either shut down or corrected the problem.</p> <p>Online log: Assertion warning or an assertion failure with a description of the operation being performed at the time of the error. Typically, an assertion warning shows that the error was internally corrected.</p> <p>Server State: Offline if the error was unrecoverable. Online if the error was corrected.</p> <p>User action: If the error was unrecoverable, start the database server and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support. No action is required if the error was internally corrected by the database server.</p>

Table C-4. Severity 5 event alarms (continued)

ID	Messages	Explanation
Class ID: 6 Event ID: 6042	Class message: Internal subsystem failure: 'message' Specific message: An internal error was reported by the database server when it detected an inconsistency with the internal buffer queues.	<p>The database server detected an inconsistency during the processing of internal buffer queues and either shut down or corrected the problem.</p> <p>Online log: Assertion warning or an assertion failure with a description of the operation being performed at the time of the error. Typically, an assertion warning shows that the error was internally corrected.</p> <p>Server State: Offline if the error was unrecoverable. Online if the error was corrected.</p> <p>User action: If the error was unrecoverable, start the database server and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support. No action is required if the error was internally corrected by the database server.</p>
Class ID: 6 Event ID: 6045	Class message: Internal subsystem failure: 'message' Specific message: Logical logging error for 'object' in 'space'	<p>The database server shut down because of an error while processing logical logs.</p> <p>Online log: Assertion failure with a description of the operation and logical log information.</p> <p>Server State: Offline.</p> <p>User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.</p>
Class ID: 6 Event ID: 6049	Class message: Internal subsystem failure: 'message' Specific message: Lock types <i>lock_type</i> and <i>lock_type</i> should never be merged	<p>The database server shut down after attempting to merge incompatible locks.</p> <p>Online log: Assertion failure with the lock types that the database server was attempting to merge.</p> <p>Server State: Offline.</p> <p>User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.</p>

Table C-4. Severity 5 event alarms (continued)

ID	Messages	Explanation
Class ID: 6 Event ID: 6050	Class message: Internal subsystem failure: 'message' Specific message: An internal error was reported by the database server when it detected some corruption in the lock free list chain.	The database server shut down after detecting corruption of an internal structure that manages an internal list of free locks. Online log: Assertion failure. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6060	Class message: Internal subsystem failure: 'message' Specific message: Thread exited with <i>number</i> buffers held	The database server shut down after detecting that a thread is holding one or more buffers. Online log: Assertion failure with the number of buffers being held by the thread. Server State: Offline. User action: Bring the database server online. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6062	Class message: Internal subsystem failure: 'message' Specific message: Internal Error - Freeing transaction entry that still holds locks!	While freeing resources associated with a transaction, the database server detected that the transaction is holding locks. In some circumstances the database server can release these locks. Online log: Assertion failure with the transaction or an assertion warning with the transaction and a statement that the database server internally corrected the problem. Server State: Offline if the error was unrecoverable. Online if the error was internally corrected. User action: If the database server shut down, start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.

Table C-4. Severity 5 event alarms (continued)

ID	Messages	Explanation
Class ID: 6 Event ID: 6067	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (Recursive exception) has caused the database server processes to terminate unexpectedly.	The database server detected recursive calls to exception handling and immediately shut down to avoid an infinite loop. Online log: Assertion failure. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6068	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (Internal exception) has caused the database server processes to terminate unexpectedly.	The database server shut down due to an unrecoverable internal error. Online log: Assertion failure with information about the exception that caused the problem. Server State: Offline. User action: Start the database server. Look at the exception information in the assertion failure file. If the exception relates to a user-defined routine, investigate and correct the user-defined routine. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6069	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (Master daemon died) has caused the database server processes to terminate unexpectedly.	The master daemon oninit process stopped and the database server shut down. This error can be caused by the termination of operating system processes. Online log: Assertion failure. Server State: Offline. User action: Start the database server. Be careful when terminating operating system processes.
Class ID: 6 Event ID: 6070	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (VP died) has caused the database server processes to terminate unexpectedly.	An oninit process stopped and the database server shut down. This error can be caused by the termination of operating system processes. Online log: Assertion failure. Server State: Offline. User action: Start the database server. Be careful when terminating operating system processes.

Table C-4. Severity 5 event alarms (continued)

ID	Messages	Explanation
Class ID: 6 Event ID: 6071	Class message: Internal subsystem failure: 'message' Specific message: ERROR: cannot fork secondary Server thread (MACH11 Shutdown)	The secondary server shut down but was unable to create a thread to shut down normally. Online log: DR: Shutting down the server. ERROR: can not fork secondary Server thread (MACH11 Shutdown) Can not run onmode -ky PANIC: Attempting to bring system down. Server State: Offline. User action: None.
Class ID: 6 Event ID: 6075	Class message: Internal subsystem failure: 'message' Specific message: A fatal internal error (KAIO) has caused the database server processes to terminate unexpectedly.	The database server shut down because of an error in the KAIO subsystem. Online log: Assertion failure with the specific operation that failed. Server State: Offline. User action: Start the database server. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 6 Event ID: 6500	Generic event for when the database server terminates unexpectedly due to an internal error condition.	An internal error occurred and the database server shut down. Online log: Assertion failure. Server State: Offline. User action: Start the database server. Examine the assertion failure file for more information about what happened. If possible, fix any problems identified and try the operation again. If the operation fails again, note all circumstances and contact IBM Software Support.
Class ID: 21 Event ID: 21004	Class message: Database server resource overflow: 'resource_name' Specific message: Physical log file overflow	The physical log file is full and needs to overflow. If this happens during recovery, the database server attempts to extend the physical log. Online log: Assertion failure if the database server is either not in recovery or is unable to extend the physical log. Assertion warning if the database server is in recovery and extends the physical log. Server State: Offline. User action: None.

Table C-4. Severity 5 event alarms (continued)

ID	Messages	Explanation
Class ID: 21 Event ID: 21006	Class message: Database server resource overflow: ' <i>resource_name</i> ' Specific message: Logical log buffer overflow detected	The database server shut down because the logical log buffer is full. Online log: Assertion failure with the log record size and the buffer size. Server State: Offline. User action: Increase the value of the LOGBUFF configuration parameter in the onconfig file. Start the database server.

RS Secondary Server Event Alarms

The following table lists the event alarms that RS secondary servers trigger. When each alarm is triggered, a message is written to the message log.

Table C-5. Event Alarms for RS Secondary Servers

ID	Severity	Specific Message	Description
Class ID: 40 Event ID: 40001	3	RSS <i>servername</i> log replay position is falling too far behind RSS source	This message displays when the log replay position for the RS secondary server is falling too far behind the primary. If this trend continues, the primary might not have a logical log file available when the RS secondary server requests it.
Class ID: 40 Event ID: 40002	3	RSS <i>servername</i> added	The RS secondary server with name <i>servername</i> was added to the cluster.
Class ID: 40 Event ID: 40003	3	RSS <i>servername</i> deleted	The RS secondary server with name <i>servername</i> was removed from the cluster.
Class ID: 40 Event ID: 40004	3	HA password for RSS <i>servername</i> changed	The password to configure the RS secondary server with name <i>servername</i> was changed.
Class ID: 40 Event ID: 40005	3	RSS <i>servername</i> is not acknowledging log transmissions	The RS secondary server with name <i>servername</i> is not acknowledging log transmissions. The primary server will not send any more logical log pages until an acknowledgment is received.

Table C-5. Event Alarms for RS Secondary Servers (continued)

ID	Severity	Specific Message	Description
Class ID: 40 Event ID: 40006	3	Error receiving a buffer from RSS <i>servername</i> shutting down	The primary server experienced an error receiving a message from the RS secondary server with name <i>servername</i> .

SD Secondary Server Event Alarms

The following table lists the event alarms that SD secondary servers trigger. When each alarm is triggered, a message is written to the message log.

Table C-6. Event Alarms for SD Secondary Servers

Class ID	Severity	Class Message	Description
Class ID: 41 Event ID: 41001	3	ERROR: Removing SDS node <i>servername</i> has timed out - removing	The SD secondary server was removed due to a time-out situation. For more information, see the SDS_TIMEOUT configuration parameter.

Appendix D. Discontinued configuration parameters

The current version of Informix does not use some configuration parameters that were used in earlier versions of the server. This appendix lists the discontinued and obsolete configuration parameters for Informix.

Table D-1 summarizes the discontinued parameters. Although these parameters are still supported, it is recommended that you do not use them. Remove these parameters from the ONCONFIG file before using the VPCLASS parameter.

Table D-1. Discontinued Configuration Parameters

Configuration Parameter	Reference
AFF_NPROCS	"AFF_NPROCS (Discontinued)" on page D-2
AFF_SPROC	"AFF_SPROC (Discontinued)" on page D-2
RA_THRESHOLD	"RA_THRESHOLD configuration parameter (discontinued)" on page D-10
BUFFERS	"BUFFERS (Discontinued)" on page D-3
FAST_RESTART_CKPT_FUZZYLOG	"FAST_RESTART_CKPT_FUZZYLOG (Discontinued)" on page D-4
FAST_RESTART_PHYSLOG	"FAST_RESTART_PHYSLOG (Discontinued)" on page D-4
LRU_MAX_DIRTY	"LRU_MAX_DIRTY (Discontinued)" on page D-6
LRU_MIN_DIRTY	"LRU_MIN_DIRTY (Discontinued)" on page D-6
LRUS	"LRUS (Discontinued)" on page D-7
NOAGE	"NOAGE (Discontinued)" on page D-7
NUMAIOVPS	"NUMAIOVPS (Discontinued)" on page D-8
NUMCPUVPS	"NUMCPUVPS (Discontinued)" on page D-8
PHYSDBS	"PHYSDBS (Discontinued)" on page D-9

Table D-2 summarizes the configuration parameters that are no longer supported.

Table D-2. Obsolete Configuration Parameters

Configuration Parameter	Reference
JDKVERSION	"JDKVERSION (Discontinued)" on page D-5
LBU_PRESERVE	"LBU_PRESERVE (Discontinued)" on page D-5
LOGSMAX	"LOGSMAX (Discontinued)" on page D-5
SB_CHECK_FOR_TEMP	"SB_CHECK_FOR_TEMP configuration parameter (discontinued)" on page D-10

Related reference:

“Database configuration parameter reference” on page 1-24

“BUFFERPOOL configuration parameter” on page 1-36

AFF_NPROCS (Discontinued)

onconfig.std value

0

units Number of CPUs

range of values

0 through number of CPUs in the computer

takes effect

When the database server shuts down and restarts

refer to

- Virtual-processor es, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*
- “AFF_SPROC (Discontinued)”
- “VPCLASS configuration parameter” on page 1-156

On multiprocessor computers that support *processor affinity*, AFF_NPROCS specifies the number of CPUs to which the database server can bind CPU virtual processors. Binding a CPU virtual processor to a CPU causes the virtual processor to run exclusively on that CPU. The database server assigns CPU virtual processors to CPUs in serial fashion, starting with the processor number that AFF_SPROC specifies.

If you specify more CPU virtual processors than there are processors, the database server starts over again at the beginning. For example, if you set AFF_NPROCS to 3 and AFF_SPROC to 5, the database server assigns two CPU virtual processors to processor 5, two CPU virtual processors to processor 6, and one CPU virtual processor to processor 7.

Important: Use VP instead of AFF_NPROCS to specify the number of CPUs. You cannot use both AFF_NPROCS and VP *cpu* in the same ONCONFIG file.

AFF_SPROC (Discontinued)

onconfig.std value

0

units CPU number

range of values

0 through (AFF_NPROCS - NUMCPUVPS + 1)

takes effect

When the database server shuts down and restarts

refer to

- Virtual-processor classes, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*
- “AFF_NPROCS (Discontinued)”
- “VPCLASS configuration parameter” on page 1-156

On multiprocessor computers that support *processor affinity*, AFF_SPROC specifies the CPU, starting with 0, on which the database server starts binding CPU virtual processors to CPUs. The AFF_NPROCS parameter specifies the number of CPUs that the database server will use. The NUMCPUVPS parameter specifies the number of CPU virtual processors to be started, and the AFF_SPROC parameter specifies the CPU on which the first virtual processor is to start. For example, if you assign eight CPUs (AFF_NPROCS = 8), and set NUMCPUVPS to 3 and AFF_SPROC to 5, the database server binds CPU virtual processors to the fifth, sixth, and seventh CPUs.

Important: Use VPCLASS instead of AFF_SPROC to specify processor affinity. You cannot use both AFF_SPROC and VPCLASS *cpu* in the same ONCONFIG file.

BUFFERS (Discontinued)

onconfig.std *value*

UNIX: 5000Windows : 2000

units Number of buffers

range of values

For 32-bit platform on UNIX: with page size equal to 2048 bytes: 100 through 1,843,200 buffers (1843200 = 1800 * 1024)

with page size equal to 4096 bytes: 100 through 921,600 buffers (921,600 = ((1800 * 1024)/4096) * 2048)

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

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

takes effect

When the database server is shut down and restarted

utilities

onstat -b or **-B** (See “**onstat -b** command: Print buffer information for buffers in use” on page 20-26.)

refer to

- Shared-memory buffer pool in the shared-memory chapter of the *IBM Informix Administrator's Guide*
- “RA_PAGES configuration parameter” on page 1-113
- Your *IBM Informix Performance Guide*

Note: Information that was specified with the BUFFERS configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see “BUFFERPOOL configuration parameter” on page 1-36.

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

In general, buffer space should range from 20 to 25 percent of physical memory. It is recommended that you calculate all other shared-memory parameters after you set buffer space ($\text{BUFFERS} * \text{system_page_size}$) to 20 percent of physical memory.

FAST_RESTART_CKPT_FUZZYLOG (Discontinued)

onconfig.std *value*

The FAST_RESTART_CKPT_FUZZYLOG parameter does not need to be in the **onconfig.std** file.

range of values

0 (default) = Disable the flushing of dirty fuzzy pages to the physical log at checkpoint.

1 = Enable the flushing of dirty fuzzy pages to the physical log at checkpoint.

takes effect

At the checkpoint that occurs after the parameter is enabled. If the total number of unflushed, dirty fuzzy pages exceeds 20 percent of the total physical log space, the pages will not be written to the physical log. If server fails after this checkpoint, crash recovery receives no performance benefit.

refer to Information on fast recovery and alternative fast restart recovery options for fuzzy operations in the *IBM Informix Administrator's Guide*.

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

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

FAST_RESTART_PHYSLOG (Discontinued)

onconfig.std *value*

The FAST_RESTART_PHYSLOG parameter does not need to be in the **onconfig.std** file.

range of values

0 (default) = Disable physical logging on fuzzy checkpoints during the roll-forward (log replay) phase of recovery.

1 = Enable physical logging on fuzzy checkpoints during the roll-forward (log replay) phase of recovery, thus decreasing recovery time.

takes effect

Immediately. If the total number of unflushed, fuzzy dirty pages exceeds 20 percent of the total physical log space, the pages will not be written to the physical log. However, if the database server fails before the next checkpoint performs, maximum fast-recovery performance does not occur because the database server did not log all of the fuzzy updates in the checkpoint intervals.

refer to Information on fast recovery and alternative fast restart recovery options for fuzzy operations in the *IBM Informix Administrator's Guide*

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

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

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

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

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

JDKVERSION (Discontinued)

onconfig.std *value*
1.4

range of values

For this release, the valid values are 1.4, 1.3, and 1.2.

takes effect

When shared memory is initialized

JDKVERSION is the major version of the JDK or JRE release. That is, the version number does not include *x* when the version is JDK 1.4.*x*.

This parameter is required if the number of JVPs (set in VPCLASS JVP) is greater than 0.

LBU_PRESERVE (Discontinued)

Informix no longer supports the LBU_PRESERVE parameter, which reserves the last logical log for ON-Archive use. ON-Archive, which has been discontinued, was the only utility that required free log space to back up a logical log.

LOGSMAX (Discontinued)

Informix no longer supports the LOGSMAX parameter.

LOGSMAX specifies the maximum number of logical-log files for a database server instance. The database server requires at least three logical-log files for operation. The maximum number of logical logs is 32,767. The LOGSMAX value must be equal to or less than the highest log file number.

LRU_MAX_DIRTY (Discontinued)

onconfig.std *value*

60.00

units Percent

range of values

0 through 100 (fractional values are allowed)

takes effect

When the database server is shut down and restarted

refer to The following topics in the shared-memory chapter of the *IBM Informix Dynamic Server Administrator's Guide*

- LRU queues
- Limiting the number of pages added to the MLRU queues

Note: Information that was specified with the LRU_MAX_DIRTY configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see “BUFFERPOOL configuration parameter” on page 1-36.

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

LRU_MIN_DIRTY (Discontinued)

onconfig.std *value*

50.00

units Percent

range of values

0 through 100 (fractional values are allowed)

takes effect

When the database server is shut down and restarted

refer to The following topics in the shared-memory chapter of the *IBM Informix Dynamic Server Administrator's Guide*:

- LRU queues
- When MLRU cleaning ends

Note: Information that was specified with the LRU_MIN_DIRTY configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see “BUFFERPOOL configuration parameter” on page 1-36.

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

LRUS (Discontinued)

onconfig.std *value*
8

if not present

If MULTIPROCESSOR is set: MAX(4, num_cpu_vps) If MULTIPROCESSOR is not set: 4

units Number of LRU queues

range of values

1 through 128

takes effect

When the database server is shut down and restarted

utilities

onstat -R (see “**onstat -R** command: Print LRU, FLRU, and MLRU queue information” on page 20-207.)

refer to

- LRU queues, in the shared-memory chapter of the *IBM Informix Dynamic Server Administrator's Guide*
- Chapter on configuration effects on memory, in your *IBM Informix Dynamic Server Performance Guide*

Note: Information that was specified with the LRUS configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see “BUFFERPOOL configuration parameter” on page 1-36.

LRUS specifies the number of LRU (least-recently-used) queues in the shared-memory buffer pool. You can tune the value of LRUS, in combination with the LRU_MIN_DIRTY and LRU_MAX_DIRTY parameters, to control how frequently the shared-memory buffers are flushed to disk.

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

NOAGE (Discontinued)

onconfig.std *value*
0

range of values

0 = Use priority aging. 1 = Disable priority aging.

takes effect

When the database server shuts down and restarts

refer to

- Preventing priority aging, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*
- “VPCLASS configuration parameter” on page 1-156

Some operating systems lower the priority of processes as the processes run over a long period of time. NOAGE, when set to 1, disables *priority aging* of CPU virtual processors by the operating system. When NOAGE is set to the default of 0, the operating system might lower the priority of CPU virtual processors, as well as

other processes, as they accumulate processing time. If your operating system supports priority aging, it is recommended that you set NOAGE to 1.

Important: It is recommended that you specify priority aging with the VPCLASS parameter instead of the NOAGE parameter. You cannot use both NOAGE and VPCLASS *cpu* in the same ONCONFIG file.

NUMAIOVPS (Discontinued)

onconfig.std value

None

if not present

If AUTO_AIOVPS is set to 1 (on), the number of AIO VPs initially started is equal to the number of AIO chunks, up to a maximum of 128.

If AUTO_AIOVPS is set to 0 (off), the number of AIO VPs started is equal to the greater of 6 or twice the number of AIO chunks, up to a maximum of 128.

units Number of AIO VPs

range of values

Integer between the value of 1 and 10,000, inclusive

takes effect

When the database server shuts down and restarts

utilities

onmode -p in “onmode -p: Add or remove virtual processors” on page 14-19

refer to

- Asynchronous I/O, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*
- “VPCLASS configuration parameter” on page 1-156

NUMAIOVPS specifies the number of virtual processors of the AIO class to run. Unless kernel asynchronous I/O is implemented, the AIO virtual processors perform all the database server disk I/O, other than I/O to the log files.

Important: It is recommended that you specify the number of AIO VPs with VPCLASS **aio** instead of NUMAIOVPS. You cannot use both NUMAIOVPS and VPCLASS **aio** in the same ONCONFIG file.

UNIX Only:

If your platform has kernel-asynchronous I/O (KAIO) turned on, the database server uses AIO virtual processors to perform I/O only to cooked chunks. The database server uses KAIO to perform all I/O to raw disk space and to the physical and logical logs. For details, see the machine notes.

NUMCPUVPS (Discontinued)

onconfig.std value

1

units Number of CPU VPs

range of values

1 through the number of CPUs

takes effect

When the database server shuts down and restarts

utilities

onmode -p in “onmode -p: Add or remove virtual processors” on page 14-19

refer to

- CPU virtual processors, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*
- “VPCLASS configuration parameter” on page 1-156

NUMCPUVPS specifies the number of virtual processors of the CPU class to run. CPU virtual processors run all threads that start as the result of a connection by a client application, as well as internal threads. In general, allocate only one CPU virtual processor on a single-processor computer or node. On a multiprocessor computer or node, do not allocate more CPU virtual processors than there are CPUs.

Important: It is recommended that you specify the number of CPU virtual processors with VPCLASS *cpu* instead of NUMCPUVPS. You cannot use both NUMCPUVPS and VPCLASS *cpu* in the same ONCONFIG file.

On UNIX, use the **onmode -p -1 CPU** command to decrease the number of CPU VPs. On Windows, you can add a CPU VP, but you cannot subtract it.

PHYSDBS (Discontinued)

onconfig.std *value*

rootdb

if not present

The dbspace that ROOTNAME specifies

units A dbspace

range of values

Up to 128 bytes. PHYSDBS must be unique, begin with a letter or underscore, and contain only letters, numbers, underscores, or \$ characters.

takes effect

When the database server is initialized

refer to

- “**onparams -p**: Change physical-log parameters” on page 16-3
- Where the physical log is located, in the chapter on what is the physical log in the *IBM Informix Administrator's Guide*
- Changing the physical-log location and size, in the chapter on managing the physical log in the *IBM Informix Administrator's Guide*

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

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

RA_THRESHOLD configuration parameter (discontinued)

Use the RA_THRESHOLD configuration parameter to specify the read-ahead threshold, which is the number of unprocessed data pages in memory that signals the database server to perform the next read-ahead.

onconfig.std *value*
16

if not present
RA_PAGES/2

units Number of data pages

range of values
0 through (RA_PAGES - 1)

takes effect
When the database server is shut down and restarted

RA_THRESHOLD is used with RA_PAGES when the database server reads during sequential scans of data records.

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

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

SB_CHECK_FOR_TEMP configuration parameter (discontinued)

Use the SB_CHECK_FOR_TEMP configuration parameter to prevent the copying of a temporary smart large object into a permanent table.

By default, you can copy temporary smart large objects into permanent tables. Smart large object data types, BLOB and CLOB, consist of two parts: the data, which is stored in an sbspace, and the handle, which is stored in a table. When you copy a temporary smart large object into a permanent table, only the BLOB or CLOB handle is copied into the permanent table. If you subsequently drop the temporary smart large object, the permanent table contains a handle that is no longer valid.

To prevent the copying of a temporary smart large object into a permanent table, set the SB_CHECK_FOR_TEMP configuration parameter to 1 in the **onconfig** file. For example, if the SB_CHECK_FOR_TEMP configuration parameter is set to 1, an INSERT INTO . . . SELECT FROM . . . statement that copies a temporary smart large object into a permanent table fails.

onconfig.std *value*
None

if not present
Permits the copying of temporary smart large objects into permanent tables.

range of values

- 0 = Permit the copying of temporary smart large objects into permanent tables. Equivalent to the configuration parameter not being set in the **onconfig** file.

- 1 = Prevent the copying of temporary smart large objects into permanent tables. The database server returns the following error messages instead of copying the handle of a temporary smart large object:
 - -9810: Smart-large-object error.
 - -12246: Smart large objects: You cannot put a temporary smart large object into a permanent table

takes effect

When the database server is shut down and restarted

Appendix E. Error Messages

This chapter lists nonnumbered messages that are printed in the database server message log and provides corrective actions.

For information on numbered messages and the unnumbered ON-Bar messages, search for the message text in the error messages file, which is located in the subdirectory for your locale under the **\$INFORMIXDIR/msg** directory. You can also search *IBM Informix Error Messages* in English at the IBM Informix Information Center at <http://publib.boulder.ibm.com/infocenter/idshelp/v117/index.jsp>.

Some of the messages included below might require you to contact Technical Support staff. Such messages are rarely, if ever, seen at customer locations.

For information on what the message log is, see installing and configuring the database server in the *IBM Informix Administrator's Guide*. For information on specifying the path to the message file, see “MSGPATH configuration parameter” on page 1-96.

How the Messages Are Ordered in This Chapter

Database server message-log messages are arranged in this chapter in alphabetical order, sorted with the following additional rules:

- The time stamp that precedes each message is ignored.
- Letter case is ignored in alphabetization.
- Spaces are ignored.
- Quotation marks are ignored.
- Leading ellipses are ignored.
- The word *the* is ignored if it is the first word in the message.
- Messages that begin with numbers or punctuation symbols appear toward the end of the list in a special section labeled “Messages: Symbols” on page E-48.
- Certain related messages are grouped together, as follows:
 - “Conversion/Reversion Messages” on page E-49
 - “Conversion and Reversion Messages for Enterprise Replication” on page E-61
 - “Dynamic Log Messages” on page E-65
 - “Sbspace Metadata Messages” on page E-67
 - “Truncate Table Messages” on page E-68

A cause and suggested corrective action for a message or group of messages follow the message text.

How to View These Messages

Use one of the following methods to view these messages:

- Online message log
To see the messages displayed as they occur, use the **tail -f online.log** command.
- **onstat -m** command

For more information, see “**onstat -l** command: Print physical and logical log information” on page 20-193.

- IBM Informix Server Administrator (ISA)

For more information, see the ISA online help.

To see the error number associated with these unnumbered messages, view the **logmessage** table in the **sysmaster** database:

```
SELECT * FROM logmessage;
```

Message Categories

Four general categories of unnumbered messages exist, although some messages fall into more than one category:

- Routine information
- Assertion-failed messages
- Administrative action needed
- Unrecoverable error detected

Technical Support uses the assertion-failed messages to assist in troubleshooting and diagnostics. The information that they report often falls into the category of *unexpected events* that might or might not develop into problems caught by other error codes. Moreover, the messages are terse and often extremely technical. They might report on one or two isolated statistics without providing an overall picture of what is happening. This information can suggest to technical support possible research paths.

Messages: A-B

Aborting Long Transaction: *tx 0xn.*

Cause

The transaction spans the log space specified by transaction high-watermark (LTXHWM), and the offending long transaction is rolling back.

Action

No additional action is needed. The address of the transaction structure in shared memory is displayed as a hexadecimal value.

Affinitied VP *mm* to phys proc *nn.*

Cause

The database server successfully bound a CPU virtual processor to a physical processor.

Action

None required.

Affinity not enabled for this server.

Cause

You tried to bind your CPU virtual processors to physical processors, but the database server that you are running does not support process affinity.

Action

Set `AFF_NPROCS` to 0, or remove the affinity setting from `VPCLASS`.

Assert Failed: Error from SBSpace cleanup thread.

Cause

The sbspace cleanup thread encountered an error while cleaning up stray smart large objects.

Action

See the action suggested in the message log file.

Most of the time, running `onspaces -cl sbspacename` on the failed sbspace succeeds in cleaning up any stray smart large objects. If you encounter an unrecoverable error, contact Technical Support.

Assert Failed: Short description of what failed Who: Description of user/session/thread running at the time Result: State of the affected database server entity Action: What action the database administrator should take See Also: DUMPDIR/af.uniqid containing more diagnostics.

Cause

This message indicates an internal error.

Action

The `af.uniqid` file in the directory specified by the `ONCONFIG` parameter `DUMPDIR` contains a copy of the assertion-failure message that was sent to the message log, as well as the contents of the current, relevant structures and/or data buffers. The information included in this message is intended for Technical Support.

Begin re-creating indexes deferred during recovery.

Cause

During recovery, indexes to be created are deferred until after recovery completes. This message indicates that the database server deferred re-creating indexes and that it is now creating the indexes. During the time that the database server re-creates the indexes, it locks the affected tables with a shared lock.

Action

None required.

Building 'sysmaster' database requires ~mm pages of logical log. Currently there are nn pages available. Prepare to back up your logs soon.

Cause

You do not currently have the approximate amount of free log space necessary to complete a build of the sysmaster database.

Action

Back up your logs.

Building 'sysmaster' database...

Cause

The database server is building the **sysmaster** database.

Action

None required.

Messages: C

Cannot Allocate Physical-log File, mm wanted, nn available.

Cause

The database server attempted to initialize shared memory with a physical-log size that exceeds the amount of contiguous space available in the dbspace (specified as PHYSDBS in ONCONFIG). Both quantities of space, wanted and available, are expressed as kilobytes.

Action

You must either reduce the size of the physical log (specified as PHYSFILE in ONCONFIG) or change the location of the physical log to a dbspace that contains adequate contiguous space to accommodate the physical log.

Cannot alter a table which has associated violations table.

Cause

The user tried to add, drop, or modify a column in a table that has a violations table associated with it.

Action

Do not change the columns in the user table.

Cannot change to mode.

Cause

Some error during fast or full recovery has prevented the system from changing to online or quiescent mode.

Action

See previous messages in the log file for information.

Cannot Commit Partially Complete Transactions.

Cause

Transactions that drop tables or indexes do not perform the drop until a COMMIT statement is processed (with a few exceptions). In these cases, a *beginning commit* log record is written, followed by the usual commit log record. If the database server fails in between the two, the fast recovery process attempts to complete the commit the next time that you initialize the database server.

If this completion of the commit fails, the database server generates the preceding message.

Action

To determine if you need to take action, examine the logical log as described in Chapter 5, "Interpreting Logical-Log Records," on page 5-1.

Cannot create a user-defined VP class with 'SINGLE_CPU_VP' non-zero.

Cause

SINGLE_CPU_VP is set to nonzero, and **onmode** was used to create a user-defined VP class.

Action

If user-defined VP classes are necessary, stop the database server, change SINGLE_CPU_VP to zero, and restart the database server.

Cannot create violations/diagnostics table.

Cause

The user issued a START VIOLATIONS TABLE statement for a target table. The database server cannot create the violations table for this target table. Any of the following situations might be the reason for this failure:

- The target table already has a violations table.
- You specified an invalid name for the violations table in the START VIOLATIONS TABLE statement. For example, if you omit the USING clause from the statement and if the number of characters in the target table plus four characters is longer than the maximum identifier length, the generated names of the violations table exceed the maximum identifier length.
- You specified a name for the violations table in the START VIOLATIONS TABLE statement that match the names of existing tables in the database.
- The target table contains columns with the names **informix_tupleid**, **informix_optype**, or **informix_recowner**. Because these column names duplicate the **informix_tupleid**, **informix_optype**, or **informix_recowner** columns in the violations table, the database server cannot create the violations table.
- The target table is a temporary table.
- The target table is serving as a violations table for some other table.

- The target table is a system catalog table.

Action

To resolve this error, perform one of the following actions:

- If the violations table name was invalid, specify a unique name for the violations table in the USING clause of the START VIOLATIONS TABLE statement.
- If the target table contains columns with the names **informix_tupleid**, **informix_optype**, or **informix_reowner**, rename them to something else.
- Choose a permanent target table that is not a system catalog table or a violations table for some other table.

Cannot insert from the violations table to the target table.

Cause

The user has issued a statement that attempts to insert rows from the violations table into the target table. For example, the user enters the following invalid statement:

```
INSERT INTO mytable SELECT * FROM mytable_vio;
```

Also, if the target table has filtering-mode constraints, you receive this error.

Action

To recover from this error, perform the following actions:

- Do not use filtering constraints.
- Stop the violations table.
- Insert rows from the violations table into a temporary table, and then insert rows from the temporary table into the target table.

Cannot modify/drop a violations/diagnostics table.

Cause

The user has tried to alter or drop a table that is serving as a violations table for another table.

Action

Do not alter or drop the violations table.

Cannot Open Dbospace *nnn*.

Cause

The database server is unable to access the specified dbospace. This message indicates a problem opening the tblspace or corruption in the initial chunk of the dbospace.

Action

Verify that the device or devices that make up the chunks of this dbospace are functioning properly and that you assigned them the correct operating-system permissions (rw-rw----). You might be required to perform a data restore.

Cannot Open Logical Log.

Cause

The database server is unable to access the logical-log files. Because the database server cannot operate without access to the logical log, you must resolve this problem.

Action

Verify that the chunk device where the logical-log files reside is functioning and has the correct operating-system permissions (rw-rw----).

Cannot Open Mirror Chunk *pathname*, **errno** = *nn*.

Cause

The database server cannot open the mirrored chunk of a mirrored pair. The chunk *pathname* and the operating-system error are returned.

Action

For more information about corrective actions, see your operating-system documentation.

Cannot Open Primary Chunk *pathname*, **errno** = *nnn*.

Cause

The primary chunk of a mirrored pair cannot be opened. The chunk *pathname* and the operating-system error are returned.

Action

For more information about corrective actions, see your operating-system documentation.

Cannot Open Primary Chunk *chunkname*.

Cause

The *initial* chunk of the dbspace cannot be opened.

Action

Verify that the chunk device is running properly and has the correct operating-system permissions (rw-rw----).

Cannot open sysams in database *name*, **iserrno** *number*.

Cause

An error occurred when the database server opened the **sysams** system table.

Action

Note the error *number* and contact Technical Support.

Cannot open sysdistrib in database *name*, iserrno *number*.

Cause

An error occurred when the database server accessed the **sysdistrib** system table.

Action

Note the error *number* and contact Technical Support.

Cannot open *system_table* in database *name*, iserrno *number*.

Cause

An error occurred when the database server opened the specified system table.

Action

Note the error *number* and contact Technical Support.

Cannot open systribody in database *name*, iserrno *number*.

Cause

An error occurred when the database server accessed the **systribody** system table.

Action

Note the error *number* and contact Technical Support.

Cannot open systribggers in database *name*, iserrno *number*.

Cause

An error occurred when the database server accessed the **systribggers** system table.

Action

Note the error *number* and contact Technical Support.

Cannot open sysxdtypes in database *name*, iserrno *number*.

Cause

An error occurred while accessing the **sysxdtypes** system table.

Action

Note the error *number* and contact Technical Support.

Cannot Perform Checkpoint, shut system down.

Cause

A thread that is attempting to restore a mirrored chunk has requested a checkpoint, but the checkpoint cannot be performed.

Action

Shut down the database server.

Cannot Restore to Checkpoint.

Cause

The database server is unable to recover the physical log and thus unable to perform fast recovery.

Action

If the database server does not come online, perform a data restore from dbspace backup.

Cannot Rollback Incomplete Transactions.

Cause

Within the fast-recovery or data-restore procedure, the logical-log records are first rolled forward. Then, open transactions that have not committed are rolled back. An open transaction could fail during the rollback, leaving some of the modifications from the open transaction in place. This error does not prevent the database server from moving to quiescent or online mode, but it might indicate an inconsistent database.

Action

To determine if any action is needed, use the onlog utility to examine the logical log.

Cannot update pagezero.

Cause

A failure occurred while the database server was trying to rewrite a reserved page during the reversion process.

Action

See previous messages in the log file for information, or contact Technical Support.

Cannot update syscasts in database *name*. Iserrno *number*.

Cause

An internal error occurred while inserting data into the **syscasts** system table.

Action

Contact Technical Support..

Can't affinity VP *mm* to phys proc *nn*.

Cause

The database server supports process affinity, but the system call to bind the virtual processor to a physical processor failed.

Action

See your operating-system documentation.

Changing the sbospace minimum extent value: old value *value1*, new value *value2*.

Cause

This informational message occurs when you issue the following command:
`onspaces -ch sbospace -Df "MIN_EXT_SIZE=value1" -y`

Action

None. For more information, see “onspaces -ch: Change sbospace default specifications” on page 19-17.

Checkpoint blocked by down space, waiting for override or shutdown.

Cause

A dbspace has gone down during a checkpoint interval. The database server is configured to wait for an override when this situation occurs.

Action

Either shut down the database server or issue an **onmode -O** command to override the down dbspace. For more information on the **onmode** utility, see Chapter 14, “The onmode utility,” on page 14-1.

Checkpoint Completed: duration was *n* seconds.

Cause

A checkpoint completed successfully.

Action

None required.

Checkpoint Page Write Error.

Cause

The database server detected an error in an attempt to write checkpoint information to disk.

Action

For additional assistance in resolving this situation, contact Technical Support.

Checkpoint Record Not Found in Logical Log.

Cause

The logical log or the chunk that contains the logical log is corrupted. The database server cannot initialize.

Action

Perform a data restore from dbspace backup.

Chunk *chunkname* added to space *spacename*.

Cause

The variables in this message have the following values:

chunkname

is the name of the chunk that the database server administrator is adding.

spacename

is the name of the storage space to which the database server administrator is adding the chunk.

Action

None required.

Chunk *chunkname* dropped from space *spacename*.

Cause

The database server administrator dropped chunk *chunkname* from space *spacename*.

Action

None required.

Chunk *number nn pathname* -- Offline.

Cause

The indicated chunk in a mirrored pair has been marked with status D and taken offline. The other chunk in the mirrored pair is operating successfully.

Action

Take steps now to repair the chunk device and restore the chunk. The chunk *number* and chunk device *pathname* are displayed.

Chunk *number nn pathname* -- Online.

Cause

The indicated chunk in a mirrored pair has been recovered and is online (marked with status 0). The chunk *number* and chunk device *pathname* are displayed.

Action

None required.

The chunk *pathname* must have READ/WRITE permissions for owner and group.

Cause

The chunk *pathname* does not have the correct owner and group permissions.

Action

Make sure that you assigned the correct permissions (-rw-rw---) to the device on which the chunk is located.

The chunk *pathname* must have *owner-ID* and *group-ID* set to **informix**.

Cause

The chunk *chunkname* does not have the correct owner and group ID.

Action

Make sure the device on which the chunk is located has the ownership. On UNIX, both owner and group should be **informix**. On Windows, the owner must be a member of the **Informix-Admin** group.

The chunk *pathname* will not fit in the space specified.

Cause

The chunk *pathname* does not fit in the space that you specified.

Action

Choose a smaller size for the chunk, or free space where the chunk is to be created.

Cleaning stray LOs in **sbspace** *sbspacename*.

Cause

The database server administrator is running **onspaces -cl sbspacename**.

Action

None required.

Completed re-creating indexes.

Cause

The database server finished re-creating the deferred indexes.

Action

None required.

Configuration has been grown to handle up to *integer* chunks.

Cause

The database server administrator increased the number of chunks to the specified value by changing CONFIGSIZE or setting MAX_CHUNKS to a higher value.

Action

None required. The change was successful.

Configuration has been grown to handle up to *integer* dbslices.

Cause

The database server administrator increased the number of dbslices to the specified value by changing CONFIGSIZE or setting MAX_DBSLICES to a higher value.

Action

None required. The change was successful.

Configuration has been grown to handle up to *integer* dbspaces.

Cause

The database server administrator increased the number of dbspaces to the specified value by changing CONFIGSIZE or setting MAX_DBSPACES to a higher value.

Action

None required. The change was successful.

Continuing Long Transaction (for COMMIT): *tx 0xn*.

Cause

The logical log has filled beyond the long-transaction high-watermark (LTXHWM), but the offending long transaction is in the process of committing. In this case, the transaction is permitted to continue writing to the logical log and is not rolled back. The address of the transaction structure in shared memory is displayed as hexadecimal value *tx 0xn*.

Action

None required.

Could not disable priority aging: *errno* = *number*.

Cause

An operating-system call failed while it was trying to disable priority aging for the CPU virtual processor. The system error *number* associated with the failure is returned.

Action

See your operating-system documentation.

Could not fork a virtual processor: errno = *number*.

Cause

The fork of a virtual processor failed. The database server returns the operating-system error *number* associated with the failure.

Action

For information on determining the maximum number of processes available per user and for the system as a whole, refer to your operating-system documentation.

Create_vp: cannot allocate memory.

Cause

The database server cannot allocate new shared memory.

Action

The database server administrator must make more shared memory available. This situation might require increasing SHMTOTAL or reconfiguring the operating system. This message is usually accompanied by other messages that give additional information.

Messages: D-E-F

Dataskip is OFF for all dbspaces.

Cause

Informational.

Action

None required.

Dataskip is ON for all dbspaces.

Cause

Informational.

Action

None required.

Dataskip is ON for dbspaces: *dbspacelist*.

Cause

Informational; DATASKIP is ON for the specified dbspaces.

Action

None required.

Dataskip will be turned {ON|OFF} for *dbspacename*.**Cause**

Informational; DATASKIP is ON or OFF for the specified dbspace.

Action

None required.

DBSERVERALIASES exceeded the maximum limit of 32**Cause**

The limit of 32 aliases was reached.

Action

Nothing. Only the first 32 will be used.

DBSPACETEMP internal list not initialized, using default.**Cause**

An error occurred while initializing a user-specified DBSPACETEMP list. Typically this condition is due to a memory-allocation failure.

Action

Check for accompanying error messages.

The DBspace/BLOBspace *spacename* is now mirrored.**Cause**

You successfully added mirroring to the indicated storage space.

Action

None required.

The DBspace/BLOBspace *spacename* is no longer mirrored.**Cause**

You have ended mirroring for the indicated storage space.

Action

None required.

Dbospace *dbspacename* for Physical-log File not found.

Cause

The dbospace *dbspacename* specified by the PHYSDBS configuration parameter does not exist. As a consequence, the database server cannot complete initialization.

Action

Use a dbospace known to exist.

***devname*: write failed, file system is full.**

Cause

Because the file system *devname* is full, the write failed.

Action

Free some space in *devname*.

Dropping temporary tblspace *0xn*, recovering *nn* pages.

Cause

During shared-memory initialization, the database server routinely searches for temporary tables that are left without proper cleanup. If the database server finds a temporary table, it drops the table and recovers the space. The database server located the specified temporary tblspace and dropped it. The value *0xn* is the hexadecimal representation of the tblspace number.

Action

None required.

Dynamically allocated new shared memory segment (size *nnnn*).

Cause

This status message informs you that the database server successfully allocated a new shared-memory segment of size *nnnn*.

Action

None required.

ERROR: NO "wait for" locks in Critical Section.

Cause

The database server does not permit a thread to own locks that might have to wait while that thread is within a critical section. Any such lock request is denied, and an ISAM error message is returned to the user.

Action

The error reported is an internal error. Contact IBM Informix Technical Support.

Error building sysmaster database. See *outfile*.

Cause

Errors were encountered in building the sysmaster database. The file *outfile* contains the result of running the script *buildsmi*.

Action

See the file *outfile*.

Error in dropping system defined type.

Cause

An internal error occurred while updating either the **sysxdtypes**, **sysctddesc**, or **sysxdttypeauth** system table.

Action

Contact Technical Support.

Error in renaming systdist.

Cause

An internal error occurred while trying to find and rename the **Informix.systdist** SPL routine.

Action

Contact Technical Support.

Error removing sysdistrib row for *tabid* = *tabid*, *colid* = *colid* in database *name*. *iserrno* = *number*

Cause

An error occurred while updating the **sysdistrib** system table.

Action

Note the error *number* and contact Technical Support.

Error writing *pathname* *errno* = *number*.

Cause

The operating system cannot write to *pathname*. *Number* is the number of the operating-system error that was returned.

Action

Investigate the cause of the operating-system error. Usually it means that no space is available for the file. It might also mean that the directory does not exist or that no write permissions exist.

Error writing shmem to file *filename* (error). Unable to create output file *filename* errno=*mm*. Error writing *filename* errno=*nn*.

Cause

The database server detected an error in an attempt to write shared memory to *filename*. The first message is followed by one of the next two. Either the attempt failed because the output file could not be created or because the contents of shared memory could not be written. The error refers to the operating-system error that prompted the attempted write of shared memory to a file. The value of *nn* is the operating-system error.

Action

See your operating-system documentation.

Fail to extend physical log space.

Cause

The attempt to extend the physical log space failed. Either the path does not exist or the permissions are incorrect.

Action

Use a path that exists. Check permissions on the current working directory. You or the system administrator must give your group execute permission on the current working directory. After your group has been given permission, retry the operation that generated this message.

Fatal error initializing CWD string. Check permissions on current working directory. Group *groupname* must have at least execute permission on '.'.

Cause

Group *groupname* does not have execute permission for the current working directory.

Action

Check permissions on the current working directory. You or the system administrator must give your group execute permission on the current working directory. After your group has been given permission, retry the operation that generated this message.

The following tables have outstanding old version data pages due to an In-Place Alter Table. Perform `UPDATE tablename SET column = column WHERE 1=1;` to clear these pages from the following tables.

Cause

Reversion to a previous version of the database server has been attempted while an in-place ALTER TABLE is in progress. The previous versions of the database server cannot handle tables that have multiple schemas of rows in them.

Action

Force any in-place alters to complete by updating the rows in the affected tables before you attempt to revert to a previous version of the database server. To do this, create a dummy update in which a column in the table is set to its own value, forcing the row to be updated to the latest schema in the process without actually changing column values. Rows are always altered to the latest schema, so a single pass through the table that updates all rows completes all outstanding in-place alters.

Fragments *dbspacename1 dbspacename2* of table *tablename* set to non-resident.

Cause

The specified fragments of *tablename* either have been set to nonresident by the SET TABLE statement.

Action

None required.

Forced-resident shared memory not available.

Cause

The database server port for your computer does not support forced-resident shared memory.

Action

None required.

Freed *mm* shared-memory segment(s) *number* bytes.

Cause

The database server sends this message to the message log after you run the **-F** option of the **onmode** utility to free unused memory. The message informs you of the number of segments and bytes that the database server successfully freed.

Action

None required.

Messages: G-H-I

gcore *pid*; mv *core.pid* *dir/core.pid*.ABORT.

Cause

This status message during a database server failure provides the name and place of each core file associated with the virtual processors.

Action

None required.

I/O *function chunk mm, pagenum nn, pagecnt aa --> errno = bb.*

Cause

An operating-system error occurred during an attempt to access data from disk space. The operating-system function that failed is defined by *function*. The chunk number and physical address of the page where the error occurred are displayed as integers. The *pagecnt* value refers to the number of pages that the thread was attempting to read or write. If an *errno* value is displayed, it is the number of the operating-system error and might explain the failure. If *function* is specified as *bad request*, some unexpected event caused the I/O attempt on an invalid chunk or page.

Action

If the chunk status changes to D, or down, restore the chunk from its mirror or repair the chunk. Otherwise, perform a data restore.

I/O error, *primary/mirror Chunk pathname -- Offline (sanity).*

Cause

The database server detected an I/O error on a primary or mirror chunk with *pathname*. The chunk was taken offline.

Action

Check that the device on which the chunk was stored is functioning as intended.

Deleted Indexes idx1 and idx 2 error message

Informix *database_server* Initialized - Complete Disk Initialized.

Cause

Disk space and shared memory have been initialized. Any databases that existed on the disk before the initialization are now inaccessible.

Action

None required.

Informix *database_server* Initialized - Shared Memory Initialized.

Cause

Shared memory has been initialized.

Action

None required.

Informix *database_server* Stopped.

Cause

The database server has moved from quiescent mode to offline mode. The database server is offline.

Action

None required.

ERROR: Insufficient available disk in the root dbspace to increase the entire Configuration save area.

Cause

The user attempted to increase the number of storage objects to a specific value by changing CONFIGSIZE or setting MAX_DBSPACES, MAX_DBSLICES, or MAX_CHUNKS to a higher value, but the database server did not have enough rootspace for the increased number of storage objects. A storage object might be a dbspace, dbslice, or chunk.

Action

Increase the size of the root dbspace or reset CONFIGSIZE, MAX_DBSPACES, MAX_DBSLICES, or MAX_DBSLICES to a lower value and restart the database server. For example, if you set MAX_CHUNKS to 32,768, but the root dbspace did not have enough space, set MAX_CHUNKS to a lower value.

Insufficient available disk in the root dbspace for the CM save area. Increase the size of the root dbspace in the ONCONFIG file and reinitialize the server.

Cause

The cause might be one of the following:

- The user attempted to increase the number of storage objects to a specific value by changing CONFIGSIZE or setting MAX_DBSPACES, MAX_DBSLICES, or MAX_CHUNKS to a higher value, but the database server did not have enough rootspace for the increased number of storage objects. A storage object might be a dbspace, dbslice, or chunk.
- The user converted to a database server version that requires slightly more rootspace, but it is not available (this case is unlikely).

Action

Take one of the following actions:

- Increase the size of the root dbspace or reset CONFIGSIZE, MAX_DBSPACES, MAX_DBSLICES, or MAX_DBSLICES to a lower value and restart the database server. For example, if you set MAX_DBSPACES to 32,768 but the root dbspace did not have enough space, set MAX_DBSPACES to a lower value.
- Increase the size of the root dbspace and reinitialize the database server.

Internal overflow of shmid's, increase system max shared memory segment size.

Cause

The database server was initializing shared memory when it ran out of internal storage for the shared-memory IDs associated with this segment.

Action

Increase the value of your maximum kernel shared-memory segment size, usually SHMMAX. For more information, see your operating-system documentation.

Messages: J-K-L-M

Listener-thread err = *error_number*: *error_message*.

Cause

A listener thread has encountered an error. This message displays the error number and message text.

Action

For the cause and corrective action, see the IBM Informix Information Center at <http://publib.boulder.ibm.com/infocenter/idshelp/v117/index.jsp>.

Lock table overflow - user id *mm* session id *nn*.

Cause

A thread attempted to acquire a lock when no locks were available. The user ID and session ID are displayed.

Action

Increase the LOCKS configuration parameter, and initialize shared memory.

Logical-log File not found.

Cause

The checkpoint record in the root dbspace reserved page is corrupted.

Action

Perform a data restore from dbspace backup.

Logical Log *nn* Complete.

Cause

The logical-log file identified by log-ID number *nn* is full. The database server automatically switches to the next logical-log file in the sequence.

Action

None required.

Logical logging *v*berror for *type:subtype* in (*failed_system*).

Cause

Logging failed. The log record that caused the error is identified as follows:

type Is the logical-log record type.

subtype
Is the logging subsystem.

failed_system
Is the name of an internal function that indicates what system failed to log.

Action

Contact Technical Support.

Log Record: log = *ll*, pos = *0xn*, type = *type:subtype(snum)*, trans = *xx*

Cause

The database server detected an error during the rollforward portion of fast recovery or logical-log restore.

The log record that caused the error is identified as follows:

ll Is the logical-log ID where the record is stored.

0xn Is the hexadecimal address position within the log.

type Is the logical-log record type.

subtype
Is the logging subsystem.

snum Is the subsystem number.

xx Is the transaction number that appears in the logical log.

Action

Contact Technical Support.

Log record (*type:subtype*) at log *nn*, *0xn* was not undone.

Cause

A log undo failed because a log is corrupt.

The log record that caused the error is identified as follows:

type Is the logical-log record type.

subtype
Is the logging subsystem.

nn Is the logical-log ID where the record is stored.

0xn Is the hexadecimal address position within the log.

Action

To determine if any action is needed, use the onlog utility to examine the logical log. Contact Technical Support.

Log record (*type:subtype*) failed, partnum *pnum* row *rid* iserrnum.

Cause

A logging failure occurred.

The log record that caused the error is identified as follows:

type Is the logical-log record type.

subtype
Is the logging subsystem.

pnum Is the part number.

rid Is the row ID.

num Is the iserror number.

Action

Contact Technical Support.

Log record (*type:subtype*) in log *nn*, offset *0xn* was not rolled back.

Cause

A log undo failed because a log is corrupt.

The log record that caused the error is identified as follows:

type Is the logical-log record type.

subtype
Is the logging subsystem.

log Is the logical-log ID where the record is stored.

offset Is the hexadecimal address position within the log.

Action

To determine if any action is needed, use the onlog utility to examine the logical log. Contact Technical Support.

Logical Recovery allocating *nn* worker threads *thread_type*.

Cause

The database server determined the number of worker threads that will be used for parallel recovery. The variable *thread_type* can assume the values ON_RECVRY_THREADS or OFF_RECVRY_THREADS.

Action

This status message requires no action. If you want a different number of worker threads allocated for parallel recovery, change the value of the ONCONFIG configuration parameter ON_RECOVERY_THREADS or OFF_RECOVERY_THREADS.

Logical Recovery Started.

Cause

Logical recovery began.

Action

This status message requires no action.

Maximum server connections *number*.

Cause

Outputs with each checkpoint message to indicate the maximum number of concurrent connections to the database server since the last restart.

Action

This message helps the customer track license usage to determine when more licenses need to be purchased. For assistance, Contact Technical Support.

Memory allocation error.

Cause

The database server ran out of shared memory.

Action

Take one of the following actions:

1. Increase swap space on the computer.
2. Check kernel shared-memory parameters for limits on shared memory.
3. Decrease the size of the memory allocated, with the **buffers** field in the BUFFERPOOL configuration parameter.
4. Increase the virtual-memory size (SHMVIRTSIZE), the size of the added segments, (SHMADD), or your total shared-memory size (SHMTOTAL).

Mirror Chunk *chunkname* added to space *spacename*. Perform manual recovery.

Cause

Fast recovery, full recovery, or an HDR secondary has recovered the add of a mirror chunk. It does not perform automatic mirror recovery, however. The administrator must do this.

Action

Use either the **onspaces** utility or ON-Monitor to attempt to recover the mirror chunks.

Mixed transaction result. (*pid=nn user=userid*).

Cause

You receive this message only when more than one database server is involved in a transaction. This message indicates that a database server, after preparing a transaction for commit, heuristically rolled back the transaction, and the global transaction completed inconsistently. The *pid* value is the user-process identification number of the coordinator process. The value of *user* is the user ID associated with the coordinator process.

Action

See the information on recovering manually from failed two-phase commit in your *IBM Informix Administrator's Guide*.

mt_shm_free_pool: pool *0xn* has blocks still used (id *nn*).

Cause

An internal error occurred during a pool deallocation because blocks are still associated with the pool.

Action

Contact Technical Support.

mt_shm_init: can't create *resident/virtual* segment.

Cause

The causes for the failure to create the resident or virtual segment are as follows: (1) the segment size is less than the minimum segment size; (2) the segment size is larger than the maximum segment size; (3) allocating another segment would exceed the allowable total shared-memory size; or (4) a failure occurred while the database server was trying to allocate the segment.

Action

If you suspect that this error was generated because of item 1 or 2 in the preceding paragraph, Contact Technical Support. To correct item 3, increase the SHMTOTAL value in your ONCONFIG configuration file. For additional information about errors generated because of item 4, see your logical-log file.

mt_shm_remove: WARNING: may not have removed all/correct segments.

Cause

When the operating system tried to remove the shared-memory segments associated with the database server, the last segment did not equal the last segment registered internally. This situation is probably due to the unexpected failure of the database server.

Action

Remove any segments that were not cleaned up.

Messages: N-O-P

Newly specified value of *value* for the pagesize in the configuration file does not match older value of *value*. Using the older value.

Cause

This message displays upon database server restart. The PAGESIZE value changed in the ONCONFIG file after the database server was initialized.

Action

The database server uses the older PAGESIZE value.

Not enough main memory.

Cause

The database server detected an error in an attempt to acquire more memory space from the operating system.

Action

For more information about shared-memory configuration and management, refer to your operating-system documentation.

Not enough logical-log files, Increase LOGFILES.

Cause

During a data restore, the value of the LOGFILES configuration must always be greater than or equal to the total number of logical-log files. At some point during the restore, the number of logical-log files exceeded the value of LOGFILES.

Action

Increase the value of LOGFILES in ONCONFIG.

Not enough physical procs for affinity.

Cause

The ONCONFIG parameters AFF_NPROCS and AFF_SPROC are not correctly set. AFF_SPROC plus AFF_NPROCS is greater than the number of physical processors on your computer or node.

Action

Reset AFF_NPROCS and AFF_SPROC, such that the value AFF_SPROC plus value of AFF_NPROCS is less than or equal to the number of physical processors.

The number of configured CPU poll threads exceeds NUMCPUVPS.

Cause

The number of in-line poll threads that you specified in the ONCONFIG configuration file exceeds the number of CPU virtual processors.

Action

Reduce the number of in-line poll threads to be less than or equal to the number of CPU virtual processors.

onconfig parameter *parameter* modified from *old_value* to *new_value*.

Cause

When the database server shared memory is reinitialized, this message documents any changes that occurred since the last initialization.

Action

None required.

oninit: Cannot have SINGLE_CPU_VP non-zero and number of CPU VPs greater than 1.

Cause

The ONCONFIG file contains VPCLASS cpu with a num= value greater than 1 and a nonzero value for SINGLE_CPU_VP. SINGLE_CPU_VP must be 0 (or omitted) when there are more than 1 CPU VPs.

Action

Correct the ONCONFIG file and restart the database server.

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

Cause

The ONCONFIG file contains a user-defined VPCLASS as well as a nonzero value for SINGLE_CPU_VP. SINGLE_CPU_VP must be 0 (or omitted) when the ONCONFIG file contains a user-defined VPCLASS.

Action

Correct the ONCONFIG file and restart the database server.

oninit: Cannot mix VPCLASS cpu and NUMCPUVPS, AFF_SPROC, AFF_NPROCS, or NOAGE parameters.

Cause

The ONCONFIG file contains both VPCLASS cpu and one or more of the other listed parameters. It cannot contain both.

Action

Correct the ONCONFIG file and restart the database server.

oninit: Cannot mix VPCLASS aio and NUMAIOVPS parameters.

Cause

The ONCONFIG file contains both VPCLASS aio and NUMAIOVPS. It cannot contain both.

Action

Correct the ONCONFIG file and restart the database server.

oninit: Fatal error in initializing ASF with 'ASF_INIT_DATA' flags asfcode = '25507'.

Cause

The **nettype** value specified in the **sqlhosts** file or registry for the database server is invalid or unsupported, or the **servicename** specified in the **sqlhosts** file or registry for the database server is invalid.

Action

Check the **nettype** and **servicename** values in the **sqlhosts** file or registry for each DBSERVERNAME and for the DBSERVERALIASES. Check the **nettype** value in each NETTYPE parameter in the ONCONFIG file.

oninit: invalid or missing name for Subsystem Staging Blobspace.

Cause

You set the configuration parameter STAGEBLOB to a blobspace that does not exist.

Action

Use the **-d** option of **onspaces** to create the blobspace specified in STAGEBLOB, and restart the database server.

Cannot alter a table which has associated violations table.

Cause

The user tried to add, drop, or modify a column in a table that has a violations table associated with it.

Action

Do not change the columns in the user table.

oninit: Too many VPCLASS parameters specified.

Cause

Too many VPCLASS parameter lines have been specified in the ONCONFIG file.

Action

Reduce the number of VPCLASS lines, if possible. If not possible, contact Technical Support.

oninit: VPCLASS *classname* bad affinity specification.

Cause

The affinity specification for the VPCLASS line is incorrect. Affinity is specified as a range:

For m , use processor m .

For m to n , use processors in the range m to n inclusive, where $m \leq n$, $m \geq 0$, and $n \geq 0$.

Action

Correct the VPCLASS parameter in the ONCONFIG file and restart the database server.

oninit: VPCLASS *classname* duplicate class name.

Cause

The VPCLASS *classname* in the ONCONFIG file has a duplicate name. VP class names must be unique.

Action

Correct the duplicate name and restart the database server.

oninit: VPCLASS *classname* illegal option.

Cause

One of the fields in the VPCLASS *classname* parameter is illegal.

Action

Correct the parameter in the ONCONFIG file and restart the database server.

oninit: VPCLASS *classname* maximum number of VPs is out of the range 0-10000.

Cause

The maximum number of VPs specified by a VPCLASS parameter line must be in the range 1 to 10,000.

Action

Correct the value and restart the database server.

oninit: VPCLASS *classname* name is too long. Maximum length is *maxlength*.

Cause

The length of the name field in VPCLASS *classname* is too long.

Action

Choose a shorter class name, correct the ONCONFIG file, and restart the database server.

oninit: VPCLASS *classname* number of VPs is greater than the maximum specified.

Cause

The initial number of VPs specified by a VPCLASS parameter is greater than the maximum specified by the same VPCLASS parameter.

Action

Correct the VPCLASS parameter and restart the database server.

oninit: VPCLASS *classname* number of VPs is out of the range 0-10000.

Cause

The initial number of VPs specified by a VPCLASS parameter line must be in the range 1 to 10,000.

Action

Correct the value and restart the database server.

onmode: VPCLASS *classname* name is too long. Maximum length is *maxlength*.

Cause

The name of a dynamically added VP class that **onmode -p** specifies is too long.

Action

Choose a shorter name, and retry the **onmode -p** command.

Optical Subsystem is running.

Cause

You set the value of the STAGEBLOB parameter in the configuration file, and the database server is communicating properly with the optical-storage subsystem.

Action

No action is required.

Optical Subsystem is not running.

Cause

You set the value of the STAGEBLOB parameter in the configuration file, but the database server cannot detect the existence of the optical-storage subsystem.

Action

Check that the optical subsystem is online.

Optical Subsystem STARTUP Error.

Cause

The database server detects that the optical-storage subsystem is running, but the database server cannot communicate with it properly.

Action

Check your optical subsystem for errors.

Online Mode.

Cause

The database server is in online mode. Users can access all databases

Action

This status message requires no action.

onspaces: unable to reset dataskip.

Cause

This error message comes from the **onspaces** utility. For some reason, the utility cannot change the specification of DATASKIP (ON or OFF) across all dbspaces in the database server instance.

Action

You are unlikely to receive this message. If the error persists after you restart the database server, Contact Technical Support.

Open transaction detected when changing log versions.

Cause

The database server detected an open transaction while it was trying to convert the data from a previous version of the database server.

Action

Conversion is not allowed unless the last record in the log is a checkpoint. You must restore the previous version of the database server, force a checkpoint, and then retry conversion.

Out of message shared memory.

Cause

The database server could not allocate more memory for the specified segment.

Action

For additional information, see the log file.

Out of resident shared memory.

Cause

The database server could not allocate more memory for the specified segment.

Action

For additional information, see the log file.

Out of virtual shared memory.

Cause

The database server could not allocate more memory for the specified segment.

Action

For additional information, see the log file.

PANIC: Attempting to bring system down.

Cause

A fatal database server error occurred.

Action

See the error that caused the panic and attempt the corrective action suggested by the error message. For additional information that might explain the failure, refer also to other messages in the message-log file.

Participant site *database_server* heuristically rolled back.

Cause

A remote site rolled back a transaction after it reached the prepared-for-commit phase.

Action

You might need to roll back the transaction on other sites and then restart it.

Physical recovery complete: *number* pages examined, *number* pages restored.

Cause

This message displays during fast recovery. The *number of pages examined* indicates the number of page images that exist in the physical log. The *number of pages restored* indicates the actual number of pages that are restored from the physical log. The number of pages restored is always less than or equal to the number examined.

The database server might physically log a page image multiple times between checkpoints. Physical recovery restores only the first logged page image.

If a page stays in the memory buffer pool, the database server physically logs it once per checkpoint, and stores one page image in the physical log. If the buffer pool is too small, a page that is being updated many times might get forced out of the buffer pool to disk and then brought back into memory for the next update. Each time the page is brought into memory, it is physically logged again, resulting in duplicate page images in the physical log.

Action

If the *number of pages examined* is much larger than the *number of pages restored*, increase the size of the buffer pool to reduce the number of duplicate before-images. For more information, see the *IBM Informix Performance Guide*.

Physical recovery started at page (*chunk:offset*).

Cause

This message displays during fast recovery. *Chunk* is the number of the chunk that contains the physical log. *Offset* is the page offset of the start of the physical log entries. Physical recovery begins restoring pages from that point.

Action

No action required. For information on fast recovery, see the *IBM Informix Administrator's Guide*.

Portions of partition partnum of table tablename in database dbname were not logged. This partition cannot be rolled forward.

Cause

Light appends occurred to the operational table since the last backup.

Action

If you want full access to data in this table, you need to alter the table to raw and then to the desired table type. This alter operation removes inconsistencies in the table that resulted from replaying non-logged operations such as light appends.

Possible mixed transaction result.

Cause

This message indicates that error -716 has been returned. Associated with this message is a list of the database servers where the result of a transaction is unknown.

Action

For information on determining if a transaction was implemented inconsistently, see the *IBM Informix Administrator's Guide*.

Prepared participant site *server_name* did not respond.

Cause

Too many attempts were made to contact remote site *server_name*. After several timeout intervals were met, the site was determined to be down.

Action

Verify that the remote site is online and that it is correctly configured for distributed transactions. Once the remote site is ready, reinitiate the transaction.

Prepared participant site *server_name* not responding.

Cause

The database server is attempting to contact remote site *server_name*. For some unknown reason, the database server cannot contact the remote site.

Action

Verify that the remote site is online and that it is correctly configured for distributed transactions.

Messages: Q-R-S

Quiescent Mode.

Cause

The database server has entered quiescent mode from some other state. On UNIX, only users logged in as **informix** or as **root** can interact with the database server. On Windows, only members of the **Informix-Admin** group can interact with the database server. No user can access a database.

Action

None required.

Read failed. Table *name*, Database *name*, **iserrno** = *number*

Cause

An error occurred reading the specified system table.

Action

Note the error number and contact Technical Support.

Recovery Mode.

Cause

The database server entered the recovery mode. No user can access a database until recovery is complete.

Action

None required.

Recreating index: '*dbname*:"*owner*".*tablename-idxname*'.

Cause

After DDL statements implicitly or explicitly create one or more new indexes, but the database server terminates abnormally before the next checkpoint, re-creation of the new indexes is deferred until after logical recovery, instead of adding each index item row by row. After logical recovery ends, the server begins a parallel index build to re-create them. This message indicates when re-creation commences for each deferred index. (But if an index was dropped before the abnormal shutdown, it will not be re-created after logical recovery, and no message referencing that index will be printed.)

Action

None required.

Rollforward of log record failed, iserrno = *nn*.

Cause

The message appears if, during fast recovery or a data restore, the database server cannot roll forward a specific logical-log record. The database server might be able to change to quiescent or online mode, but some inconsistency could result. For further information, see the message that immediately precedes this one. The *iserrno* value is the error number.

Action

Contact IBM Informix Technical Support.

Root chunk is full and no additional pages could be allocated to chunk descriptor page.

Cause

The root chunk is full.

Action

To free space in the root chunk, take one of the following actions:

- Drop and re-create the **sysmaster** database.
- Move user tables from the root dbspace to another dbspace.
- Refragment tables.

scan_logundo: subsys *ss*, type *tt*, iserrno *ee*.

Cause

A log undo failed because log type *tt* is corrupt.

The variables in this message have the following values:

ss Is the subsystem name.

tt Is the logical-log record type.

ee Is the iserror number.

Action

Examine the logical log with the onlog utility to determine if any action is needed. Contact Technical Support.

Session completed abnormally. Committing *tx id 0xm*, flags *0xn*.

Cause

Abnormal session completion occurs only when the database server is attempting to commit a transaction that has no current owner, and the transaction develops into a long transaction. The database server forked a thread to complete the commit.

Action

None required.

Session completed abnormally. Rolling back *tx id 0xm, flags 0xn*.

Cause

Abnormal session completion occurs only when the database server is attempting to commit a distributed transaction that has no current owner, and the transaction develops into a long transaction. The database server forked a thread that rolled back the transaction.

Action

None required.

semctl: errno = *nn*.

Cause

When the database server initialized a semaphore, an error occurred. The operating-system error is returned.

Action

See your operating-system documentation.

semget: errno = *nn*.

Cause

An allocation of a semaphore set failed. The operating-system error is returned.

Action

See your operating-system documentation.

shmat: *some_string* os_errno: *os_err_text*.

Cause

An attempt to attach to a shared-memory segment failed. The system error number and the suggested corrective action are returned.

Action

Review the corrective action (if given), and determine if it is reasonable to try. For more information, refer to your operating-system documentation.

shmctl: errno = *nn*.

Cause

An error occurred while the database server tried to remove or lock a shared-memory segment. The operating-system error number is returned.

Action

See your operating-system documentation.

shmdt: errno = *nn*.

Cause

An error occurred while the database server was trying to detach from a shared-memory segment. The operating-system error number is returned.

Action

See your operating-system documentation.

shmenv sent to *filename*.

Cause

The database server wrote a copy of shared memory to the specified file as a consequence of an assertion failure.

Action

None.

shmget: *some_str* os_errno: key shmkey: *some_string*.

Cause

Either the creation of a shared-memory segment failed, or an attempt to get the shared-memory ID associated with a certain key failed. The system error number and the suggested corrective action are returned.

Action

Consult your operating-system documentation.

Shutdown (onmode -k) or override (onmode -O).

Cause

A dbpspace has gone down during a checkpoint interval. The database server is configured to wait for an override when this situation occurs.

When the checkpoint actually happens, the following message appears: Checkpoint blocked by down space, waiting for override or shutdown.

Action

Either shut down the database server or issue an **onmode -O** command to override the down dbpspace. For more information on the **onmode** utility, see Chapter 14, "The onmode utility," on page 14-1.

Shutdown Mode.

Cause

The database server is in the process of moving from online mode to quiescent mode.

Action

None required.

Space *spacename* added.

Cause

The database server administrator added a new storage space *spacename* to the database server.

Action

None required.

Space *spacename* dropped.

Cause

The database server administrator dropped a storage space *spacename* from the database server.

Action

None required.

Space *spacename* -- Recovery Begins(*addr*).

Cause

This informational message indicates that the database server is attempting to recover the storage space.

The variables in this message have the following values:

spacename

Is the name of the storage space that the database server is recovering.

addr

Is the address of the control block.

Action

None required.

Space *spacename* -- Recovery Complete(*addr*).

Cause

This informational message indicates that the database server recovered the storage space.

The variables in this message have the following values:

spacename

Is the name of the storage space that the database server has recovered.

addr Is the address of the control block.

Action

None required.

Space *spacename* -- Recovery Failed(*addr*).**Cause**

This informational message indicates that the database server was unable to recover the storage space.

The variables in this message have the following values:

spacename

Is the name of the storage space that the database server failed to recover.

addr Is the address of the control block.

Action

None required.

sysmaster database built successfully.**Cause**

The database server successfully built the sysmaster database.

Action

None required.

Successfully extend physical log space**Cause**

The physical log space was successfully extended to the file `plog_extend.servernum` under the designated path.

Action

None required.

Messages: T-U-V
This ddl operation is not allowed due to deferred constraints pending on this table and dependent tables.**Cause**

This error is returned when you attempt to start a violations table and constraints are in deferred mode.

Note: No error is returned if you start a violations table and then later set the constraints to deferred. However, the violations get undone immediately rather than written into the deferred constraint buffer. For more information, see the *IBM Informix Guide to SQL: Syntax*.

Action

If you would like to start a violations table, you must either change the constraint mode to immediate or commit the transaction.

This type of space does not accept log files.

Cause

Adding a logical-log file to a blobspace or sbpace is not allowed.

Action

Add the logical-log file to a dbspace. For more information, see “onparams -a -d *dbspace*: Add a logical-log file” on page 16-2.

TIMER VP: Could not redirect I/O in initialization, errno = nn.

Cause

The operating system could not open the null device or duplicate the file descriptor associated with the opening of that device. The system error number is returned.

Action

See your operating-system documentation.

Too Many Active Transactions.

Cause

During a data restore, there were too many active transactions. At some point during the restore, the number of active transactions exceeded 32 kilobytes.

Action

None.

Too many violations.

Cause

The number of violations in the diagnostics table exceeds the limit that is specified in the MAX VIOLATIONS clause of the START VIOLATIONS TABLE statement. When a single statement on the target table (such as an INSERT or UPDATE statement) inserts more records into the violations table than the limit that is specified by the MAX VIOLATIONS clause, this error is returned to the user who issued the statement on the target table.

Action

To resolve this error, perform one of the following actions:

- Omit the MAX VIOLATIONS clause in the START VIOLATIONS TABLE statement when you start a violations table. Here, you are specifying no limit to the number of rows in the violations table.
- Set MAX VIOLATIONS to a high value.

Transaction Not Found.

Cause

The logical log is corrupt. This situation can occur when a new transaction is started, but the first logical-log record for the transaction is not a BEGWORK record.

Action

Contact Technical Support.

Transaction heuristically rolled back.

Cause

A heuristic decision occurred to roll back a transaction after it completed the first phase of a two-phase commit.

Action

None required.

Transaction table overflow - user id *nn*, process id *nn*.

Cause

A thread attempted to allocate an entry in the transaction table when no entries in the shared-memory table were available. The user ID and process ID of the requesting thread are displayed.

Action

Try again later.

Unable to create output file *filename* errno = *nn*.

Cause

The operating system cannot create output file *filename*. The *errno* is the number of the operating-system error returned.

Action

Verify that the directory exists and has write permissions.

Unable to extend *nn* reserved pages for *purpose* in root chunk.

Cause

The operating system cannot extend to *nn* reserved pages for *purpose* in root chunk. (The value *purpose* can be either Checkpoint/Log, DBSpace, Chunk, or Mirror Chunk.)

Action

Reduce the ONCONFIG parameter for the resource cited; bring the database server up and free some space in the primary root chunk. Then reattempt the same operation.

Unable to initiate communications with the Optical Subsystem.

Cause

The optical driver supplied by the optical-drive vendor has indicated that the drive is not accessible.

Action

Check driver installation and cabling between the computer and the drive.

Unable to start SQL engine.

Cause

The database server encountered an out-of-memory condition.

Action

No action is necessary.

Unable to open tblspace *nn*, iserrno = *nn*.

Cause

The database server cannot open the specified tblspace. (The value *nn* is the hexadecimal representation of the tblspace number.)

Action

See the ISAM error message number *nn*, which should explain why the tblspace cannot be accessed. The error message appears in *IBM Informix Error Messages* at the IBM Informix Online Documentation site at: <http://www.ibm.com/developerworks/db2/zones/informix/>.

The value of pagesize *pagesize* specified in the config file is not a valid pagesize. Use 2048, 4096 or 8192 as the value for PAGESIZE in the onconfig file and restart the server.

Cause

This message displays upon disk initialization. The value of PAGESIZE that was specified in the ONCONFIG file is not a valid value.

Action

Restart the database server with a valid PAGESIZE value.

Violations table is not started for the target table.

Cause

If you issue a STOP VIOLATIONS TABLE statement for which no violations table is started, you receive this message.

Action

To recover from this error, you must start a violations table for the target table.

Violations table reversion test completed successfully.

Cause

This message is recorded in the **logmessage** table in the **sysmaster** database when the **revtestviolations.sh** script has completed successfully (no open violations tables were found).

Action

No action is necessary. For more information on **revtestviolations.sh**, see the *IBM Informix Migration Guide*.

Violations table reversion test failed.

Cause

When the database server finds an open violations table, it reports errors 16992 and 16993 in the **logmessage** table in the **sysmaster** database and aborts the reversion process.

Action

When this message appears, you must issue the STOP VIOLATIONS TABLE FOR *table_name* command for each open violations table. After you close all open violations tables, you can restart the reversion process.

Violations table reversion test start.

Cause

This message is recorded in the **logmessage** table in the **sysmaster** database when the **revtestviolations.sh** script is executed.

Action

No action is necessary. For more information on **revtestviolations.sh**, see the *IBM Informix Migration Guide*.

Violations tables still exist.

Cause

This message is recorded in the **logmessage** table in the **sysmaster** database when an open violations table is found.

Action

When this message appears, you must issue the STOP VIOLATIONS TABLE FOR *table_name* command for each open violations table. After you close all open violations tables, you can restart the reversion process.

Virtual processor limit exceeded.

Cause

You configured the database server with more than the maximum number of virtual processors allowed (1000).

Action

To reduce the number of virtual processors, decrease the values of VPCLASS, NUMCPUVPS, NUMAIOVPS, or NETTYPE in your ONCONFIG file.

VPCLASS *classname* name is too long. Maximum length is *maxlength*.

Cause

This message indicates an internal error.

Action

Contact Technical Support.

VPCLASS *classname* duplicate class name.

Cause

This message indicates an internal error.

Action

Contact Technical Support.

VPCLASS *classname* Not enough physical procs for affinity.

Cause

The physical processors in the affinity specification for the VP class *classname* do not exist or are offline. The problem might be with the VPCLASS parameter for cpu class VPs or with the AFF_SPROC and AFF_NPROCS parameters.

Action

Make sure the named processors are online. Correct the affinity specification for the named VP class. Restart the database server.

Messages: W-X-Y-Z

WARNING: aio_wait: errno = *nn*.

Cause

While the database server was waiting for an I/O request to complete, it generated error number *nn* on an operation that it was attempting to execute.

Action

Contact Technical Support for assistance.

WARNING: Buffer pool size may cause database server to get into a locked state. Recommended minimum buffer pool size is *num* times maximum concurrent user threads.

Cause

There are not enough buffers in the buffer pool. The database server could use all available buffers and cause a deadlock to occur.

Action

Change the **buffers** field in the BUFFERPOOL parameter in the ONCONFIG file to the number that this message recommends. For more information on the BUFFERPOOL parameter, see “BUFFERPOOL configuration parameter” on page 1-36..

warning: Chunk time stamps are invalid.

Cause

A sanity check is performed on chunks when they are first opened at system initialization. The chunk specified did not pass the check and will be brought offline.

Action

Restore the chunk from a dbspace backup or its mirror.

Warning: *name_old* is a deprecated onconfig parameter. Use *name_new* instead. See the release notes and the Informix Administrator's Reference for more information.

Cause

A deprecated ONCONFIG parameter was used. This message displays the first time that you use a deprecated parameter. The shorter form of the message displays thereafter.

Action

Use the suggested alternative ONCONFIG parameter.

Warning: *name_old* is a deprecated onconfig parameter. Use *name_new* instead.

Cause

A deprecated ONCONFIG parameter was used.

Action

Use the suggested alternative ONCONFIG parameter.

Warning: Unable to allocate requested big buffer of size *nn*.

Cause

The internal memory allocation for a big buffer failed.

Action

Increase either virtual memory size (SHMVIRTSIZE), the size of the added segments (SHMADD), or your total shared-memory size (SHMTOTAL).

You are turning off smart large object logging.

Cause

These changes will become the new sbspace default values. Changes have been made to the sbspace. The onspaces utility will read and update 100 smart large objects at a time and commit each block of 100 smart large objects as a single transaction. This utility might take a long time to complete.

Action

This informational message occurs when you issue the following command:

```
onspaces -ch sbspace -Df "LOGGING=OFF" -y
```

For more information, see “onspaces -ch: Change sbspace default specifications” on page 19-17.

Messages: Symbols

***HH:MM:SS Informix database server Version R.VV.PPPPP
Software Serial Number RDS#YYYYYYY.***

Cause

This message indicate the start-up of the database server, after the initialization of shared memory.

Action

No action is required.

***argument:* invalid argument.**

Cause

This internal error indicates that an invalid argument was passed to an internal routine.

Action

Contact Technical Support.

***function_name:* cannot allocate memory.**

Cause

The database server cannot allocate memory from internal shared-memory pool.

Action

Increase either virtual-memory size (SHMVIRTSIZE), the size of the added segments (SHMADD), or your total shared-memory size (SHMTOTAL).

Conversion/Reversion Messages

These messages might display during database server conversion or reversion.

Messages: A-C

Cannot revert constraint with id *id* (in syschecks).

Cause

The database has a constraint that was defined in a version more recent than the one to which you are reverting.

Action

Drop the specified constraint and retry reversion.

Cannot revert new fragment expression for index *index*, tabid *id*.

Cause

The index fragmentation was defined in a version more recent than the one to which you are reverting.

Action

Drop the problem index-fragmentation scheme and retry reversion.

Cannot revert new table fragment expression for *table* with id *id*.

Cause

The fragmentation of this table was defined in a version more recent than the one to which you are reverting.

Action

Drop the problem table fragmentation scheme and retry reversion.

Cannot update page zero.**Cause**

Attempt to write page zero failed.

Action

Contact Technical Support.

Checking database *name* for revertibility.**Cause**

Indicates that start of the reversion checks on the specified database.

Action

None required.

Conversion of pre 7.3 in-place alter started *status*.**Cause**

The database server is converting data structures for in-place alters to the new format.

Action

None required.

Conversion of pre 9.2 database tblspaces *status*.**Cause**

The database server is converting tblspaces to the new format.

Action

None required.

The conversion of the database *name* has failed.**Cause**

Indicates that the conversion of the specified database has failed.

Action

Connect to the database. This action triggers conversion of the database. If it fails, the relevant error message appears. Contact Technical Support.

Converting database *name*...**Cause**

This message appears at the start of conversion of each database in the system.

Action

None required.

Converting in-place alters to new format.

Cause

The database server is converting data structures for in-place alters to the new format.

Action

None required.

Converting 'onpload' database...

Cause

Printed in **online.log** at the beginning of **onpload** conversion.

Action

None required.

Converting partition header from version 7.x.

Cause

The database server is converting the partition header page to the new format that contains the chunk number and offset.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

Converting partition header page *address*.

Cause

The database server is converting the partition header page to the new format that contains the chunk number and page offset.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

Converting partition header pages *status*.

Cause

This message tracks the progress of the conversion of the partition header pages. The status is identified as follows:

- started
- succeeded

- FAILED

Action

If the status is started or succeeded, no action is required.

If conversion of the partition header pages failed, restart the database server. It will attempt to continue converting where it left off in the restartable conversion phase. If this action fails, diagnose the problem, restore from tape, fix the problem, and retry conversion.

Converting partition keys to 9.2.

Cause

The database server is converting the partition keys to the Version 9.2 format.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

Converting partition name for *databasename:tablename*.

Cause

The database server is converting the partition name for the *databasename:tablename*.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

Messages: D-F

The database *name* has been converted successfully.

Cause

Indicates successful completion of the conversion of the specified database.

Action

None required.

Database *name* is not revertible...

Cause

The database has failed one of the reversion checks and is not revertible.

Action

Take action to correct the error displayed as a separate message.

Database *name* is revertible...**Cause**

The database has passed all reversion checks and is revertible to the specified version.

Action

None required.

Database *name*: Must drop trigger (id = *id_number*).**Cause**

The database contains a trigger that was created in a version more recent than the one to which you are converting.

Action

Drop the trigger with the specified trigger identification number and then attempt reversion.

Database *name* SUCCESSFULLY reverted...**Cause**

Indicates the success of reversion of the specified database.

Action

None required.

... dropping sysmaster database.**Cause**

The database server is dropping sysmaster database during the reversion process.

Action

No action is required.

The dummy updates failed while converting database *name*. This may imply data corruption in the database. If so, restore the original database with the tape backup. For more information, see *output_file*.

Cause

During conversion of a database from a version earlier than Version 9.2, dummy update statements are run against the system tables in the database being converted. This message indicates failure in running one of these update statements.

Action

To retry the dummy updates, run the dummy update script for your old database server version. For instructions, refer to the *IBM Informix Migration Guide*.

If data corruption occurred, restore the original database with the tape backup. For more information, see the *IBM Informix Backup and Restore Guide*.

**The dummy updates succeeded while converting database *name*.
Cause**

During conversion of a database from a version earlier than Version 9.2, dummy update statements are run against the system tables in the database being converted. This message indicates successful completion of these updates.

Action

None required.

**Error in slow altering a system table.
Cause**

An internal error occurred while performing reversion.

Action

Contact Technical Support.

**External conversion aborted due to incompatible sysmaster database.
Cause**

The **sysmaster** database was not converted to the current database server version. A current **sysmaster** database is needed for external conversion to complete.

Action

Drop the **sysmaster** database and reboot the database server. It will build a new **sysmaster** database and relaunch external conversion automatically.

Messages: I-P

**Internal server error.
Cause**

An unexpected error occurred during database reversion.

Action

Contact Technical Support.

**Must drop long identifiers in table *name* in database *name*
Cause**

Identifiers greater than 18 bytes in length are not supported in the database server version to which you are reverting.

Action

Make sure that all long identifiers in the system are either dropped or renamed before you attempt reversion.

Must drop new database (*name*) before attempting reversion.

Iserrno *error_number*

Cause

The system contains a database that was created in a more recent version of the database server.

Action

Drop the new database and attempt reversion.

Must drop new user defined statistics in database *name*, iserrno *number*

Cause

Some distributions in the **sysdistrib** system table use user-defined statistics. This feature is not supported in the version to which you are reverting.

Action

Ensure that no user-defined statistics are present or used in the system and then attempt reversion.

ON-Bar conversion completed successfully.

Cause

ON-Bar conversion completed successfully.

Action

None.

ON-Bar conversion failed see /tmp/bar_conv.out.

Cause

ON-Bar conversion failed.

Action

For failure details, see /tmp/bar_conv.out.

ON-Bar conversion start:

Cause

ON-Bar conversion script is now running.

Action

None.

ON-Bar reversion completed successfully.

Cause

ON-Bar reversion was completed successfully.

Action

None.

ON-Bar reversion failed see /tmp/bar_rev.out.

Cause

ON-Bar reversion failed.

Action

For failure details, see /tmp/bar_rev.out.

ON-Bar reversion start:

Cause

ON-Bar reversion script is now running.

Action

None.

ON-Bar reversion test completed successfully.

Cause

ON-Bar reversion test was completed successfully.

Action

None.

ON-Bar reversion test start:

Cause

ON-Bar reversion test script is now running.

Action

None.

'onpload' conversion completed successfully.

Cause

Displayed in **online.log** at the successful completion of **onpload** conversion.

Action

None required.

'onpload' conversion failed. For details, look in \$INFORMIXDIR/etc/conpload.out.

Cause

Conversion of the **onpload** database failed.

Action

Find out the cause of failure from `$INFORMIXDIR/etc/conpload.out`. Fix the problem before you reattempt conversion.

...'onpload' reversion completed successfully.

Cause

Printed in `online.log` at the successful completion of reversion.

Action

None required.

...'onpload' reversion failed. For details, look in \$INFORMIXDIR/etc/revpload.out.

Cause

Reversion of the `onpload` database failed.

Action

Find the cause of failure in `$INFORMIXDIR/etc/revpload.out`. Fix the problem before you reattempt reversion.

'onpload' reversion test completed successfully.

Cause

Printed in `online.log` if the `onpload` database is revertible.

Action

None required.

'onpload' reversion test start:

Cause

Printed in `online.log` at the beginning of `onpload` reversion testing.

Action

None required.

The pload database contains load/unload jobs referring to long table names, column names, or database names. These jobs will not work as expected until they are redefined.

Cause

Printed during `onpload` reversion testing if the `onpload` database contains references to long table names, column names, or database names. But the reversion will complete.

Action

Redefine the load and unload jobs in the `onpload` database that have references to long identifiers.

Messages: R-W

...reverting 'onpload' database.

Cause

Printed in **online.log** at the beginning of **onpload** reversion.

Action

None required.

Reverting partition header from version 9.2.

Cause

The database server is reverting the partition header page to the old format that contains the physical address.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

Reverting partition header page *address*.

Cause

The database server is reverting the partition header page to the old format that contains the physical address.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

Reverting partition header pages *status*.

Cause

The database server is reverting the partition header pages to the old format. The status is identified as follows:

- started
- succeeded
- FAILED

Action

If reversion of the partition header pages started or succeeded, no action is required. If reversion of the partition header pages failed, restore from a tape backup, diagnose and fix the problem, and retry conversion.

Reverting partition keys to pre 9.2.

Cause

The database server is reverting the partition keys to the pre-Version 9.2 format.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

Reverting partition *name* for *databasename:tablename*.

Cause

The database server is reverting the partition name for *databasename:tablename*.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action

None required.

... reverting reserved pages.

Cause

The database server is reverting reserved pages.

Action

No action is required.

... reverting tables that underwent In-Place Alter.

Cause

The database server is reverting tables that underwent in-place alter.

Action

No action is required.

R-tree error message conversion completed successfully.

Cause

R-tree error message conversion was completed successfully.

Action

None required

R-tree error message conversion failed. (See /tmp/contree.out or %TMP%\contree.out)

Cause

R-tree error message conversion failed.

Action

See `/tmp/conR-tree.out` and `/tmp/R-tree.databases`.

R-tree error message conversion started.**Cause**

R-tree error message conversion script is now running.

Action

None required.

Reversion canceled.**Cause**

The reversion process was canceled because of errors encountered.

Action

Correct the cause of the errors, and restart reversion.

Reversion complete. Install IBM Informix database server *version* before restarting.**Cause**

The reversion process was completed successfully.

Action

You must install the older database version.

Reversion of database *name* FAILED**Cause**

Indicates the failure of reversion of the specified database.

Action

None required.

...reverting 'syscdr' database.**Cause**

Printed in `online.log` at the beginning of Enterprise Replication reversion.

Action

None required.

...starting reversion of database *name*.**Cause**

Indicates the start of actual reversion of the specified database.

Action

None required.

There is a semi-detached index in this table, which cannot be reverted. Drop this index, and retry reversion.

Cause

A semi-detached index on this table cannot be reverted.

Action

To see the list of all semi-detached indexes, refer to the database server message log. These indexes cannot be reverted. To continue reversion, drop these semi-detached indexes and retry reversion. If needed, you will need to re-create these indexes after reversion is complete.

Unable to read reserved page *chunk:offset - reserved_page*.

Cause

Both disk pages in a given reserved page pair are bad. On the disk page, *chunk* represents the chunk number and *offset* represents the page offset for the chunk.

Action

Contact Technical Support.

WARNING: Target server version must have a certified Storage Manager installed after conversion/reversion and before bringing up server.

Cause

ON-Bar is being converted or reverted. The user must ensure that a storage manager, certified with the target database server version, is installed.

Action

None.

Conversion and Reversion Messages for Enterprise Replication

Use the **concdr.sh** script on UNIX or the **concdr.bat** script on Windows to convert Enterprise Replication and the **syscdr** database to a format for an earlier version of the database server. Use the **revcdr.sh** script on UNIX or the **revcdr.bat** script on Windows to revert Enterprise Replication and the **syscdr** database to an earlier version. These scripts write conversion and reversion messages for Enterprise Replication to the following locations:

- Output of the **concdr.sh** or **concdr.bat** script, which is standard output by default
- **concdr.out** file
- Output of the **revcdr.sh** or **revcdr.bat** script, which is standard output by default
- **revcdr.out** file
- **revtestcdr.out** file

You can find the **concdr.out**, **revcdr.out**, and **revtestcdr.out** files in **\$INFORMIXDIR/etc** on UNIX or **%INFORMIXDIR%\etc** on Windows. For more information on converting and reverting Enterprise Replication, see the *IBM Informix Migration Guide*.

CDR reversion test completed successfully.

Cause

The **syscdr** database is revertible.

Action

None required.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

CDR reversion test failed; for details look in \$INFORMIXDIR/etc/revtestcdr.out.

Cause

Enterprise Replication is not revertible.

Action

For more information, look at the messages in **revtestcdr.out**. Fix the reported problem before you attempt reversion.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

Enterprise Replication is not ready for conversion. The Control and TRG send queues should be empty for conversion/reversion to proceed.

Cause

There are elements in the control and Transaction Send Queue (also called TRG) send queues. The database server sends replicated data to the TRG queue before sending it to the target system.

Action

Wait for these queues to empty before you attempt either conversion or reversion. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Prints this message to **concdr.out** during conversion or to **revcdr.out** during reversion.

Enterprise Replication is not ready for conversion. The syscdr database should NOT contain old-style group definitions for conversion to succeed.

Cause

The **syscdr** database *should not* contain old-style group definitions for conversion to succeed.

Action

Use the **cdr delete group** command to delete the old-style groups before attempting conversion. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Prints this message to **concdr.out**.

Enterprise Replication should be in a stopped state for conversion/reversion to proceed.

Cause

Enterprise Replication should be in a stopped state for conversion or reversion to proceed.

Action

Stop Enterprise Replication. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Prints this message to **concdr.out** during conversion or to **revcdr.out** during reversion.

Reversion of 'syscdr' failed; for details look in \$INFORMIXDIR/etc/revcdr.out.

Cause

The reversion of the syscdr database failed.

Action

Find the cause of failure in the **revcdr.out** file, then fix the problem before you attempt reversion.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

Starting CDR reversion test...

Cause

This message displays at the beginning of Enterprise Replication reversion testing.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

Action

None required.

Starting 'syscdr' conversion...

Cause

This message displays when you run the **concdr.sh** or **concdr.bat** script to convert the **syscdr** database to a format for an earlier version of the database server.

Action

None required.

Prints the output of the **concdr.sh** or **concdr.bat** script to standard output.

Starting 'syscdr' reversion...

Cause

This message displays when you run the **revcdr.sh** or **revcdr.bat** script to revert the **syscdr** database to an earlier version.

Action

None required.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

'syscdr' conversion completed successfully.

Cause

This message displays after you complete converting Enterprise Replication and the **syscdr** database a format for an earlier version of the database server.

Action

None required.

Prints the output of the **concdr.sh** or **concdr.bat** script to standard output.

'syscdr' conversion failed. For details, look in \$INFORMIXDIR/etc/concdr.out.

Cause

Conversion of the **syscdr** database failed.

Action

If conversion fails, resolve the problem reported in **concdr.out**. Restore the **syscdr** database from backup and reattempt conversion.

Prints the output of the **concdr.sh** or **concdr.bat** script to standard output.

Syscdr should NOT contain new replicate sets for reversion to succeed.

Cause

The new replicate sets in the **syscdr** database are not compatible with older versions.

Action

Use the **cdr delete replicateset** command to delete the replicate sets. Then rerun the **revcdr.sh** or **revcdr.bat** script to reattempt reversion.

Prints this message to **revtestcdr.out**.

Syscdr should not contain replicates defined with the --floatieee option for reversion to succeed.

Cause

Replicates have been defined with the **--floatieee** option. You cannot revert these replicates to the older version.

Action

Use the **cdr delete replicateset** command to delete replicates defined with the **--floatieee** option, then reattempt reversion.

Prints this message to **revtestcdr.out**.

Dynamic Log Messages

Dynamically added log file *logid* to DBspace *dbspace_number*.

Cause

The next active log file contains records of an open transaction. Whenever the database server adds a log dynamically, it logs this message. Example: Dynamically added log file 38 to DBspace 5.

Action

Complete the transaction as soon as possible.

Log file *logid* added to DBspace *dbspace_number*.

Cause

Whenever the administrator adds a log file manually, the database server logs this message. Example: Log file 97 added to DBspace 2.

Action

None required.

Log file number *logid* has been dropped from DBspace *dbspace_number*.

Cause

When you drop a newly-added log file, the database server logs this message.
Example: Log file number 204 has been dropped from DBspace 17.

Action

None required.

Log file *logid* has been pre-dropped.

Cause

When you drop a used log file, it is marked as deleted (status **D**) and cannot be used again. After you perform a level-0 backup, the database server drops this log file and can reuse the space. Example: Log file 12 has been pre-dropped.

Action

To delete the log file, perform a level-0 backup of all storage spaces.

Pre-dropped log file number *logid* has been deleted from DBspace *dbspace_number*.

Cause

After a backup, the database server deletes a pre-dropped log file and logs this message. Example: Pre-dropped log file number 12 has been deleted from DBspace 3.

Action

None required.

ALERT: Because the oldest logical log (*logid*) contains records from an open transaction (*transaction_address*), the server is attempting to dynamically add a log file. But there is no space available. Please add a DBspace or chunk. Then complete the transaction as soon as possible.

Cause

If the database server is unable to dynamically add a log file because the instance is out of space, it logs this message.

Action

Add a dbspace or chunk to an existing dbspace. Then complete the transaction as soon as possible.

ALERT: The oldest logical log (*logid*) contains records from an open transaction (*transaction_address*). Logical logging will remain blocked until a log file is added. Add the log file with the onparams -a command, using the -i (insert) option, as in: onparams -a -d *dbspace* -s *size* -i. Then complete the transaction as soon as possible.

Cause

If the DYNAMIC_LOGS parameter is set to 1, the database server prompts the administrator to add log files manually when they are needed.

Action

Use the onparams -a command with the -i option to add the log file after the current log file. Then complete the transaction as soon as possible.

Log file *logid* has been pre-dropped. It will be deleted from the log list and its space can be reused once you take level-0 archives of all BLOBspaces, Smart BLOBspaces and non-temporary DBspaces.

Cause

When you drop a used log file, it is marked as deleted (status D) and cannot be used again, and onparams prints this message.

Action

To delete the log file, perform a level-0 backup of all storage spaces.

Sbospace Metadata Messages

Allocated *number* pages to Metadata from chunk *number*.

Cause

The database server freed the specified number of pages from the reserved area and moved them to the metadata area of chunk *number*.

Action

None required.

Allocated *number* pages to Userdata from chunk *number*.

Cause

The database server freed the specified number of pages from the reserved area and moved them to the user-data area of chunk *number*.

Action

None required.

Freeing reserved space from chunk *number* to Metadata.

Cause

The metadata area in chunk *number* is full. The database server is trying to free space from the reserved area to the metadata area.

Action

None required.

Freeing reserved space from chunk *number* to Userdata.

Cause

The user-data area in chunk *number* is full. The database server is trying to free space from the reserved area to the user-data area.

Action

None required.

Truncate Table Messages

The table cannot be truncated if it has an open cursor or dirty readers.

Cause

You must have exclusive access to the table.

Action

Wait for dirty readers to complete or close all the open cursors and reissue the TRUNCATE TABLE command.

The table cannot be truncated. It has at least one non-empty child table with referential constraints.

Cause

You cannot truncate a table if it has child tables with referential constraints and at least one row.

Action

Empty the child tables before you truncate this table.

Appendix F. Limits in IBM Informix

The following sections list selected capacity limits and system defaults for IBM Informix.

Limitations on UNIX Operating Systems

System-Level Parameter Limits (UNIX)

System-Level Parameters	Maximum Capacity per Computer System
IBM Informix systems per computer (Dependent on available system resources)	255
Maximum number of accessible remote sites	Machine specific
Maximum virtual shared memory segment (SHMVIRTSIZE)	2GB (32-bit platforms) or 4TB (64-bit platforms)
Maximum address space	1.7GB if boot.ini file not modified to 3GB 2.7GB if boot.ini file is modified to 3GB

Table-Level Parameter Limits (UNIX)

Table-Level Parameters (based on 2K page size)	Maximum Capacity per Table
Data rows per page	255
Data rows per fragment	4,277,659,295
Data pages per fragment	16,775,134
Data bytes per fragment (excludes Smart Large Objects (BLOB, CLOB) and Simple Large Objects (BYTE, TEXT) created in Blobspaces)	33,818,671,136
Binary Large Object BLOB/CLOB pages	4*2*40
Binary Large Objects TEXT/BYTE bytes	4*2*40
Row length	32,767
Number of columns	32K
Key parts per index	16
Columns per functional index	102 (for C UDRs) 341 (for SPL or Java UDRs)
Maximum bytes per index key (for a given page size):	2K page size = 387 4K page size = 796 8K page size = 1615 12K page size = 2435 16K page size = 3254
Maximum size of an SQL statement	64K

Access capabilities (UNIX)

Access Capabilities	Maximum Capacity per System
Maximum databases per Informix system	21 million
Maximum tables per Informix system	477,102,080
Maximum active users per Informix (minus the minimum number of system threads)	32K user threads
Maximum active users per database and table (also limited by the number of available locks, a tunable parameter)	32K user threads
Maximum number of open databases in a session	8 databases
Maximum number of open tables per Informix system	Dynamic allocation
Maximum number of open tables per user and join	Dynamic allocation
Maximum locks per Informix system and database	Dynamic allocation
Maximum number of page cleaners	128
Maximum number of partitions per dbspace	4K page size: 1048445, 2K page size: 1048314 (based on 4-bit bitmaps)
Maximum number of recursive synonym mappings	16
Maximum number of tables locked with LOCK TABLE per user	32
Maximum number of cursors per user	Machine specific
Maximum Enterprise Replication transaction size	4 TB
Maximum dbspace size	4 TB with 2K page size 8 TB with 4K page size
Maximum sbspace size	4 TB with 2K page size 8 TB with 4K page size
Maximum chunk size	4 TB
Maximum number of chunks	32,766
Maximum number of 2K pages per chunk	2 billion
Maximum number of open Simple Large Objects (applies only to TEXT and BYTE data types)	20
Maximum number of B-tree levels	20
Maximum amount of decision support memory	Machine specific
Maximum size of an Informix instance	8 PB
Utility support for large files	17 billion GB
Maximum number of storage spaces (dbspaces, blobspaces, sbspaces, or extspaces)	2047

IBM Informix System Defaults (UNIX)

Table lock mode	Page
Initial extent size	8 pages
Next extent size	8 pages
Read-only isolation level (with database transactions)	Committed Read

Read-only isolation level (ANSI-compliant database)	Repeatable Read
---	-----------------

ON-Monitor Statistics (UNIX)

Number of displayed user threads	1000
Number of displayed chunks	1000
Number of displayed dbspaces	1000
Number of displayed databases	1000
Number of displayed logical logs	1000

Limitations on Windows Operating Systems

System-Level Parameter Limits (Windows)

System-Level Parameters	Maximum Capacity per Computer System
IBM Informix systems per computer (Dependent on available system resources)	255
Maximum number of accessible remote sites	Machine specific
Maximum virtual shared memory segment (SHMVIRTSIZE)	2GB (32-bit platforms) or 4TB (64-bit platforms)
Maximum address space	1.7 GB if boot.ini file not modified to 3 GB 2.7 GB if boot.ini file is modified to 3 GB

Table-Level Parameter Limits (Windows)

Table-Level Parameters (based on 2K page size)	Maximum Capacity per Table
Data rows per page	255
Data rows per fragment	4,277,659,295
Data pages per fragment	16,775,134
Data bytes per fragment (excludes Smart Large Objects (BLOB, CLOB) and Simple Large Objects (BYTE, TEXT) created in Blobspaces)	33,818,671,136
Binary Large Object BLOB/CLOB pages	4 TB
Binary Large Objects TEXT/BYTE bytes	4 TB
Row length	32,767
Number of columns	32 K
Key parts per index	16
Columns per functional index	102 (for C UDRs) 341 (for SPL or Java UDRs)

Table-Level Parameters (based on 2K page size)	Maximum Capacity per Table
Maximum bytes per index key (for a given page size):	2K page size = 387 4K page size = 796 8K page size = 1615 12K page size = 2435 16K page size = 3254
Maximum size of an SQL statement	64 K

Access Capabilities (Windows)

Access Capabilities	Maximum Capacity per System
Maximum databases per IBM Informix system	21 million
Maximum tables per IBM Informix system	477,102,080
Maximum active users per IBM Informix (minus the minimum number of system threads)	32K user threads
Maximum active users per database and table (also limited by the number of available locks, a tunable parameter)	32K user threads
Maximum number of open databases in a session	8 databases
Maximum number of open tables per IBM Informix system	Dynamic allocation
Maximum number of open tables per user and join	Dynamic allocation
Maximum locks per IBM Informix system and database	Dynamic allocation
Maximum number of page cleaners	128
Maximum number of recursive synonym mappings	16
Maximum number of tables locked with LOCK TABLE per user	32
Maximum number of cursors per user	Machine specific
Maximum Enterprise Replication transaction size	4 TB
Maximum dbspace size	8 TB
Maximum sbspace size	8 TB
Maximum chunk size	4 TB
Maximum number of chunks	32,766
Maximum number of 2K pages per chunk	2 billion
Maximum number of open Simple Large Objects (applies only to TEXT and BYTE data types)	20
Maximum number of B-tree levels	20
Maximum amount of decision support memory	Machine specific
Maximum size of a IBM Informix instance	8 PB
Utility support for large files	17 billion GB
Maximum number of storage spaces (dbspaces, blobspaces, sbspaces, or extspaces)	2047
Maximum number of partitions per dbspace	4K page size: 1048445, 2K page size: 1048314 (based on 4-bit bitmaps)

IBM Informix System Defaults (Windows)

Table lock mode	Page
Initial extent size	8 pages
Next extent size	8 pages
Read-only isolation level (with database transactions)	Committed Read
Read-only isolation level (ANSI-compliant database)	Repeatable Read

Appendix G. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features for IBM Informix products

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Informix products. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

Tip: The information center and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features by using the keyboard instead of the mouse.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

You can view the publications in Adobe Portable Document Format (PDF) by using the Adobe Acrobat Reader.

IBM and accessibility

See the *IBM Accessibility Center* at <http://www.ibm.com/able> for more information about the IBM commitment to accessibility.

Dotted decimal syntax diagrams

The syntax diagrams in our publications are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number 3.1) are mutually exclusive

alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is read as 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 refers to a separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used.

However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data-area, you know that you can include more than one data area or you can include none. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
 2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
 3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.
- + Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line 6.1+ data-area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. As for the * symbol, you can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy,

modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

- V option 6-1
- version option 6-1
- .conf.dbservername file A-5
- .inf.servicename file A-7
- .informix file A-6
- .infos.dbservername file
 - defined A-7
 - regenerating 14-23
- .infxdirs file A-7
- .jvpprops file A-8
- onstat utility
 - g act option 20-48
 - g all option 20-49
 - g arc option 20-52
 - g ath option 20-54
 - g buf option 20-55
 - g cat option 20-57
 - g cdr option 20-59
 - g cmsm option 20-69
 - g con option 20-71

Numerics

- 64-bit addressing
 - buffer pool 1-38
 - memory 1-128

A

- ac_config.std file A-4
- ac_msg.log file A-4
- Accessibility G-1
 - dotted decimal format of syntax diagrams G-1
 - keyboard G-1
 - shortcut keys G-1
 - syntax diagrams, reading in a screen reader G-1
- ACCESSTIME tag 19-13
- add bufferpool argument 21-14
- add chunk argument 21-15
- add log argument 21-16
- add memory argument 21-17
- add mirror argument 21-18
- ADDCHK logical-log record 5-3
- ADDDBS logical-log record 5-3
- Adding
 - CPU virtual processors 14-19, 14-21
- ADDITEM logical-log record 5-3
- ADDLOG logical-log record 5-3
- ADMIN_MODE_USERS configuration parameter 1-25
- ADMIN_USER_MODE_WITH_DBSA configuration parameter 1-26
- admin() functions 21-1
 - add bufferpool argument 21-14
 - add chunk argument 21-15
 - add log argument 21-16
 - add memory argument 21-17
 - add mirror argument 21-18
 - alter chunk argument 21-19
 - alter logmode argument 11-1, 21-20

- admin() functions (*continued*)
 - alter plog argument 21-21
 - archive fake argument 21-21
 - argument size 21-2
 - arguments by functional category 21-4
 - cdr argument 21-22
 - check data argument 21-23
 - check extents argument 21-24
 - check partition argument 21-24
 - checkpoint argument 21-25
 - clean sbspace argument 21-26
 - compress argument 21-126
 - compression purge_dictionary argument 21-132
 - create blobspace argument 21-26
 - create blobspace from storagepool argument 21-28
 - create chunk argument 21-29
 - create chunk from storagepool argument 21-30
 - create database argument 21-31
 - create dbspace argument 21-32
 - create dbspace from storagepool argument 21-33
 - create sbspace argument 21-34
 - create sbspace from storagepool argument 21-35
 - create sbspace with accesstime argument 21-36
 - create sbspace with log argument 21-37
 - create tempdbspace argument 21-38
 - create tempdbspace from storagepool argument 21-39
 - create tempsbspace argument 21-39
 - create tempsbspace from storagepool argument 21-41
 - create_dictionary argument 21-126
 - drop blobspace argument 21-43
 - drop blobspace to storagepool argument 21-44
 - drop chunk argument 21-44
 - drop chunk to storagepool argument 21-45
 - drop database argument 21-46
 - drop dbspace argument 21-46
 - drop dbspace to storagepool argument 21-47
 - drop log argument 21-47
 - drop sbspace argument 21-49
 - drop sbspace to storagepool argument 21-49
 - drop tempdbspace argument 21-50
 - drop tempdbspace to storagepool argument 21-50
 - drop tempsbspace to storagepool argument 21-51
 - estimate_compression argument 21-126
 - command output 21-131
 - file status 21-51
 - for compressing data 21-125, 21-126
 - fragment argument 21-126
 - ha make primary argument 21-52
 - ha rss add argument 21-54
 - ha rss argument 21-53
 - ha rss change argument 21-55
 - ha rss delete argument 21-56
 - ha sds clear argument 21-56
 - ha sds primary argument 21-57
 - ha sds set argument 21-58
 - ha set idxauto argument 21-59
 - ha set ipl argument 21-59
 - ha set primary argument 21-60
 - ha set secondary argument 21-61
 - ha set standard argument 21-61
 - ha set timeout argument 21-62

- admin() functions (*continued*)
 - message log delete 21-63
 - message log rotate 21-64
 - message log truncate 21-64
 - modify chunk extend argument 21-65
 - modify chunk extendable argument 21-66
 - modify chunk extendable off argument 21-67
 - modify space expand argument 21-68
 - modify space sp_sizes argument 21-69
 - onbar argument 21-92
 - onmode and a arguments 21-71
 - onmode and c arguments 21-72
 - onmode and C arguments 21-73
 - onmode and d arguments 21-74
 - onmode and D arguments 21-76
 - onmode and e arguments 21-76
 - onmode and F arguments 21-77
 - onmode and j arguments 21-78
 - onmode and l arguments 21-79
 - onmode and m arguments 21-79
 - onmode and M arguments 21-80
 - onmode and n arguments 21-81
 - onmode and O arguments 21-81
 - onmode and p arguments 21-82
 - onmode and Q arguments 21-83
 - onmode and r arguments 21-84
 - onmode and S arguments 21-85
 - onmode and W arguments 21-85
 - onmode and wf arguments 21-87
 - onmode and wm arguments 21-88
 - onmode and Y arguments 21-90
 - onmode and z arguments 21-91
 - onmode and Z arguments 21-91
 - onmode, wm, and AUTO_LRU_TUNING arguments 21-89
 - onmsync argument 21-93
 - ontape archive argument 21-95
 - print error argument 21-96
 - print file info argument 21-96
 - print partition argument 21-97
 - rename space argument 21-98
 - repack argument 21-126
 - repack_offline argument 21-126
 - reset sysadmin argument 21-99
 - restart listen argument 21-100
 - return codes 21-2
 - scheduler argument 21-101
 - scheduler lmm disable argument 21-104
 - scheduler lmm enable argument 21-101
 - set chunk argument 21-105
 - set dataskip argument 21-106
 - set index compression argument 21-106
 - set onconfig memory argument 21-107
 - set onconfig permanent argument 21-108
 - set sbospace accesstime argument 21-109
 - set sbospace avg_lo_size argument 21-110
 - set sbospace logging argument 21-111
 - set sql tracing argument 21-111
 - set sql tracing database argument 21-113
 - set sql tracing session argument 21-114
 - set sql tracing user argument 21-115
 - set sql user tracing argument 21-115
 - shrink argument 21-126
 - start listen argument 21-116
 - start mirroring argument 21-117
 - stop listen argument 21-117
 - stop mirroring argument 21-118

- admin() functions (*continued*)
 - storagepool add argument 21-119
 - storagepool delete argument 21-121
 - storagepool modify argument 21-122
 - storagepool purge argument 21-124
 - syntax rules 21-1
 - table argument 21-126
 - uncompress argument 21-126
 - uncompress_offline argument 21-126
- Administration mode 1-25, 14-17
- af.xxx file A-4
- AFF_NPROCS configuration parameter
 - defined D-2
 - VP D-2
- AFF_SPROC configuration parameter
 - defined D-3
 - VPCLASS D-3
- alarm scripts
 - foreground operations C-2
- ALARMPROGRAM configuration parameter 1-27, C-1
- Allocating unbuffered disk space 19-5, 19-7
- ALLOCGENPG log record 5-3
- ALLOW_NEWLINE configuration parameter 1-28
- ALRM_ALL_EVENTS configuration parameter 1-29
- alter chunk argument 21-19
- alter logmode argument 11-1, 21-20
- alter plog argument 21-21
- ALTERDONE log record 5-3
- Alternate dbservername 1-45
- ALTSPCOLSNEW log record 5-3
- ALTSPCOLSOLD log record 5-3
- ANSI/ISO-compliant database property 2-14
- archive fake argument 21-21
- Archiving
 - after renaming spaces 19-26
- Assertion failure
 - af.xxx file A-4
 - DUMPCNT configuration parameter 1-67
 - DUMPCORE configuration parameter 1-67
 - DUMPSHMEM configuration parameter 1-69
 - gcore file A-6
 - shmem.xxx file A-10
- Assertion-failed messages E-2
- Asynchronous I/O
 - cooked chunks D-8
- Audit records
 - sysadinfo table 2-7
 - sysaudit table 2-7
- AUTO_AIOVPS configuration parameter 1-29
- AUTO_CKPTS configuration parameter 1-30
- AUTO_LRU_TUNING
 - changing dynamically 14-27
- AUTO_LRU_TUNING configuration parameter 1-30
- AUTO_READAHEAD configuration parameter 1-31
- AUTO_REPREPARE configuration parameter 1-32
- AUTO_STAT_MODE configuration parameter 1-33, 1-66
- AVG_LO_SIZE tag 19-13

B

- B-tree
 - functional index 4-21
 - key-value locking 4-20
 - structure 4-16
- B-tree scanner 1-35, 14-5
 - controlling with SQL administration API commands 21-73
 - onmode -C command 21-73

- Backups
 - adding log files 16-2
 - after creating a storage space 19-5, 19-7, 19-16
 - automatic log 1-27, C-1
 - boot file A-5
 - changing physical log 16-4
 - displaying contents 13-1
 - dropping log files 16-2
 - external 14-4
 - ixbar file A-5
- BADIDX logical-log record 5-3
- BAR_ACT_LOG file 11-1
- bar_act.log file A-4
- bar_action table 2-6
- bar_instance table 2-6
- bar_object table 2-6
- bar_server table 2-6
- BATCHEDREAD_INDEX configuration parameter 1-34
- BATCHEDREAD_TABLE configuration parameter 1-34
- BEGCOM logical-log record 5-3
- BEGIN logical-log record 5-3
- beginlg field 20-217
- BEGPREP logical-log record 5-3
- BEGWORK logical-log record 5-3
- BFRMAP logical-log record 5-3
- Big-remainder page 4-14
- Binding CPU virtual processors D-2
- Bitmap page
 - blob space 4-23
 - tblspace tblspace 4-6
- bitvector field 5-3
- BLDCL logical-log record 5-3
- bldutil.sh script 2-2, A-5
- Blobpage
 - average fullness statistics 8-3, 8-13
 - size, blobpage 4-21
 - size, specifying 4-21, 19-5
 - structure
 - and storage 4-21, 4-23
 - dbspace blobpage 4-22
- Blobspaces
 - adding chunk 19-2
 - bit-map page 4-23
 - blobpage structure 4-23
 - blob space mirror chunk, structure 4-3
 - blob space structure 4-21
 - creating 19-5
 - dropping blob spaces 19-20
 - dropping chunk 19-19
 - free-map page
 - defined 4-23
 - location in blob space 4-21
 - role in blobpage logging 4-23
 - tracked by bitmap 4-23
 - maximum number 19-5, 19-7, 19-16
 - mirroring, ending 19-24
 - mirroring, starting 19-22
 - naming conventions 19-5
 - page types 4-23
 - restriction
 - dropping 19-20
 - simple-large-object storage 4-22
- Blocking
 - by checkpoints 20-61
 - database server 14-4
- BLOCKTIMEOUT configuration parameter 1-35
- BMAP2TO4 logical-log record 5-3
- BMAPFULL logical-log record 5-3
- BSPADD logical-log record 5-3
- BTCPYBCK logical-log record 5-3
- BTMERGE logical-log record 5-3
- BTSCANNER configuration parameter 1-35
- BTS Huffl logical-log record 5-3
- BTSPLIT logical-log record 5-3
- Buffer pools 19-10
 - 64-bit addressing 1-38
 - adding a pool 16-4
 - creating a pool 16-4
 - smart large objects 1-38
- Buffered
 - disk space examples 19-2
- Buffered-logging database property 2-14
- BUFFERING tag
 - in -Df option 19-13
- BUFFERPOOL configuration parameter 1-36
- Buffers
 - shown with onstat -b 20-26
 - access-level flag bits 20-27
 - page-type codes 20-28
- BUFFERS discontinued configuration parameter D-3
- buffers field in the BUFFERPOOL configuration parameter 1-38
- buildsmi script
 - error log message E-17
 - failure A-5
 - initializing database server 2-1
- buildsmi_out file A-5
- buildsmi.out file A-5

C

- Case-insensitive database property 2-14
- cdr argument 21-22
- CDR logical-log record 5-3
- CHALLOC
 - logical-log record 5-3
 - record subtype (SBLOB) 5-14
- Change
 - physical log
 - size and location 16-3
- Change Data Capture (CDC) API 21-126
- Changing
 - sbspace attributes 19-17
- changing to 14-17
- CHCOMBINE
 - logical-log record 5-3
 - record subtype, SBLOB 5-14
- check data argument 21-23
- check extents argument 21-24
- check partition argument 21-24
- CHECKALLDOMAINSFORUSER configuration parameter 1-40
- Checkpoint
 - CKPTINTVL configuration parameter 1-41
 - disabling I/O errors 1-104
 - history 20-61
 - statistics 2-8
 - warm restores performance 1-104
- checkpoint argument 21-25
- CHFREE
 - logical-log record 5-3
 - record subtype, SBLOB 5-14
- CHKADJUP log record 5-3
- CHPHYLOG log record 5-3

- CHRESERV logical-log record 5-3
- CHSPLIT
 - logical-log record 5-3
 - record subtype, SBLOB 5-14
- chunks
 - avoid overwriting 19-27
- Chunks
 - changing mirroring status 19-26
 - checking for overlap 8-12
 - free list, checking with oncheck 8-3, 8-10
 - free-list page 4-2, 4-3, 4-4
 - initial chunk of dbspace 4-1
 - initial mirror offset 1-95
 - Logical log
 - checking consistency 8-12
 - maximum number 19-2
 - monitoring 2-8
 - Physical log
 - checking consistency 8-12
 - structure
 - additional dbspace chunk 4-3
 - initial dbspace chunk 4-2
 - mirror chunk 4-3
 - using a symbolic link for the pathname 1-95, 1-118
- CINDEX logical-log record 5-3
- Cipher
 - encryption 1-71
- CKPOINT logical-log record 5-3
- CKPTINTVL configuration parameter 1-41
- clean sbpace argument 21-26
- CLEANERS configuration parameter 1-41
- Cleaning system resources 9-1, 9-2
- Client
 - killing sessions (onmode -z) 14-29
 - results of connection D-9
 - specifying dbservername 1-47
 - USEOSTIME configuration parameter 1-155
- Clone a snapshot of a database server 18-1
- CLR logical-log record 5-3
- CLUSIDX logical-log record 5-3
- COARSELOCK log record 5-3
- Cold restore
 - number of recovery threads 1-103
- COLREPAI logical-log record 5-3
- command_history table 3-8, 21-1, 21-2
- Commands
 - onstat -g cat 20-57
 - onstat -g cdr 20-58
 - onstat -g ddr 20-74
 - onstat -g dss 20-84
- COMMIT logical-log record 5-3
- Compactness of index page 1-79
- compliance with standards xxxv
- Compressing
 - Compressing
 - fragments 21-126
 - Compression
 - dictionary 21-126
 - fragments 21-125
 - tables 21-125, 21-126
- Compression
 - dictionaries 20-131
 - fragment SQL administration API arguments 21-126
 - printing progress 20-79
 - table SQL administration API arguments 21-126
 - compression purge_dictionary argument 21-132
 - Compressiondeleting dictionaries 21-132
 - Compressionestimating benefit 21-131
 - Compressionprinting dictionary information 20-131
 - COMTAB logical-log record 5-3
 - COMWORK log record 5-3
 - concdr.out file E-61
 - concdr.sh script A-1, A-5, E-61
 - Concurrent I/O
 - enabling 1-54
 - Concurrent IO 1-54
 - Configurable page size 1-36
 - Configuration file
 - and genoncfg utility 7-1
 - displaying settings 1-2
 - preparing 1-2
 - processing A-9
 - warning about modifying onconfig.std 1-2, A-8
 - Configuration parameter
 - ENCRYPT_MAC 1-73
 - ENCRYPT_MACFILE 1-74
 - configuration parameter settings 1-35
 - configuration parameters
 - changing dynamically 14-25
 - changing with omode -wf or -wm 14-25
 - WSTATS 1-163
 - Configuration parameters
 - ADMIN_MODE_USERS 1-25
 - ADMIN_USER_MODE_WITH_DBSA 1-26
 - AFF_NPROCS D-2
 - AFF_SPROC D-3
 - ALARMPROGRAM 1-27, C-1
 - ALLOW_NEWLINE 1-28
 - ALRM_ALL_EVENTS 1-29
 - attributes 1-25
 - AUTO_AIOVPS 1-29
 - AUTO_CKPTS 1-30
 - AUTO_LRU_TUNING 1-30
 - AUTO_READAHEAD 1-31
 - AUTO_REPREPARE 1-32
 - AUTO_STAT_MODE 1-33, 1-66
 - BATCHEDREAD_INDEX 1-34
 - BATCHEDREAD_TABLE 1-34
 - BLOCKTIMEOUT 1-35
 - BTSCANNER 1-35
 - BUFFERPOOL 1-36
 - characteristics 1-25
 - CHECKALLDOMAINSFORUSER 1-40
 - CKPTINTVL 1-41
 - CLEANERS 1-41
 - CONSOLE 1-42
 - CONVERSION_GUARD 1-42
 - current default values 1-2
 - database 1-25
 - DATASKIP 1-43, 21-106
 - DB_LIBRARY_PATH 1-44
 - DBC_CREATE_PERMISSION 1-44
 - DBSERVERALIASES 1-45, 14-7, 21-74
 - DBSERVERNAME 1-47, 14-7, 21-74
 - DBSPACETEMP 1-48
 - DD_HASHMAX 1-50
 - DD_HASHSIZE 1-51
 - DEADLOCK_TIMEOUT 1-51
 - DEF_TABLE_LOCKMODE 1-52
 - DEFAULTESCCHAR 1-53
 - DELAY_APPLY 1-53
 - DIRECT_IO 1-54
 - DIRECTIVES 1-55
 - DISABLE_B162428_XA_FIX 1-56

Configuration parameters (continued)

discontinued D-1
 DRAUTO 1-57
 DRDA_COMMBUFSIZE 1-57
 DRIDXAUTO 1-59
 DRINTERVAL 1-59
 DRLOSTFOUND 1-60
 DRTIMEOUT 1-60
 DS_HASHSIZE 1-61, 1-64
 DS_MAX_QUERIES 1-61, 14-11
 DS_MAX_SCANS 1-62, 14-11
 DS_NONPDQ_QUERY_MEM 1-63
 DS_POOLSIZE 1-64
 DS_TOTAL_MEMORY 1-65, 14-11
 DUMPCNT 1-67
 DUMPCORE 1-67
 DUMPDIR 1-67
 DUMPGCORE 1-68
 DUMPSHMEM 1-69
 DYNAMIC_LOGS 1-69
 EILSEQ_COMPAT_MODE 1-70
 ENABLE_SNAPSHOT_COPY 1-70
 ENCRYPT_CIPHERS 1-71
 ENCRYPT_HDR 1-72
 ENCRYPT_SMX 1-75
 ENCRYPT_SWITCH 1-75
 EXPLAIN_STAT 1-76
 EXT_DIRECTIVES 1-76
 EXTSHMADD 1-77
 FAILOVER_CALLBACK 1-78
 FAILOVER_TX_TIMEOUT 1-78
 FAST_RESTART_CKPT_FUZZYLOG D-4
 FAST_RESTART_PHYSLOG D-4
 FASTPOLL 1-79
 FILLFACTOR 1-79, 4-20
 FULL_DISK_INIT 1-80
 HA_ALIAS 1-81
 HETERO_COMMIT 1-81
 IFX_EXTEND_ROLE 1-82
 IFX_FOLDVIEW 1-82
 instance 1-25
 JDKVERSION D-5
 JVPDEBUG A-8
 JVPLOGFILE A-8
 LIMITNUMSESSIONS 1-83
 LISTEN_TIMEOUT 1-84
 LOCKS 1-84
 LOG_INDEX_BUILDS 1-87, 21-59
 LOG_STAGING_DIR 1-87
 LOGBUFF 1-85
 LOGFILES 1-86
 LOGSIZE 1-88
 LOW_MEMORY_MGR 1-89
 LOW_MEMORY_RESERVE 1-90
 LTXEHW 1-91
 LTXHWM 1-92
 MAX_FILL_DATA_PAGES 1-92
 MAX_INCOMPLETE_CONNECTIONS 1-93
 MAX_PDQPRIORITY 1-94, 14-11
 MIRROR 1-95
 MIRROROFFSET 1-95
 MIRRORPATH 1-95
 MSG_DATE 1-96
 MSGPATH 1-96
 MULTIPROCESSOR 1-97
 NET_IO_TIMEOUT_ALARM 1-97
 NETTYPE 1-98

Configuration parameters (continued)

no longer used D-1
 NS_CACHE 1-101
 NUMAIOVPS D-8
 NUMCPUVPS D-9
 NUMFDSERVERS 1-102
 OFF_RECVRY_THREADS 1-103
 ON_RECVRY_THREADS 1-104
 ONDBSPACEDOWN 1-104, 14-19, 21-81
 ONLIDX_MAXMEM 1-105
 OPCACHEMAX 1-106
 OPT_GOAL 1-106, 1-107
 OPTCOMPIND 1-106
 PC_HASHSIZE 1-108
 PC_POOLSIZE 1-109
 PHYSBUFF 1-109
 PHYSDBS D-9
 PHYSFILE 1-110
 PLOG_OVERFLOW_PATH 1-111
 PN_STAGELOB_THRESHOLD 1-111
 PRELOAD_DLL_FILE 1-112
 QSTATS 1-112
 RA_PAGES 1-113
 reference 1-25
 REMOTE_SERVER_CFG 1-113
 REMOTE_USERS_CFG 1-114
 RESIDENT 1-115
 RESTARTABLE_RESTORE 1-116
 RESTORE_POINT_DIR 1-117
 ROOTOFFSET 1-118
 ROOTPATH 1-117, 1-118
 ROOTSIZE 1-119
 RSS_FLOW_CONTROL 1-119
 RTO_SERVER_RESTART 1-120
 SBSPACENAME 1-122, 1-148
 SBSPACETEMP 1-123, 19-12
 SDS_ENABLE 1-124
 SDS_LOGCHECK 1-124
 SDS_PAGING 1-125
 SDS_TEMPDBS 1-125
 SDS_TIMEOUT 1-127
 SECURITY_LOCALCONNECTION 1-127
 SEQ_CACHE_SIZE 1-128
 SERVERNUM 1-128
 setting decision-support with onmode 14-11
 SHMADD 1-128, 21-71
 SHMBASE 1-129
 SHMNOACCESS 1-130
 SHMTOTAL 1-130
 SHMVIRT_ALLOCSEG 1-131
 SHMVIRT_SIZE 1-132
 SINGLE_CPU_VP 1-134
 SMX_COMPRESS 1-135
 SP_AUTOEXPAND 1-136
 SP_THRESHOLD 1-136
 SP_WAITTIME 1-137
 SQL_LOGICAL_CHAR 1-138
 SQLTRACE 1-139
 SSL_KEYSTORE_LABEL 1-140
 STACKSIZE 1-141
 STAGELOB 1-142
 STATCHANGE 1-142
 STMT_CACHE 1-142
 STMT_CACHE_HITS 1-143, 14-24
 STMT_CACHE_NOLIMIT 1-144
 STMT_CACHE_NUMPOOL 1-145
 STMT_CACHE_SIZE 1-145

Configuration parameters *(continued)*

- STOP_APPLY 1-145
- STORAGE_FULL_ALARM 1-147
- SYSALARMPROGRAM 1-147
- SYSSBSPACENAME 1-148
- TBLSPACE_STATS 1-149
- TBLTBLFIRST 1-150
- TBLTBLNEXT 1-150
- TEMPTAB_NOLOG 1-151
- TXTIMEOUT 1-151, 14-29
- UNSECURE_ONSTAT 1-152
- UPDATABLE_SECONDARY 1-152
- use
 - Data-replication screen 15-3
 - Initialization screen 15-3
 - PDQ screen 15-3
- USELASTCOMMITTED 1-153
- USEOSTIME 1-155
- USTLOW_SAMPLE 1-155
- viewing Enterprise Replication settings 20-59
- VP_MEMORY_CACHE_KB 1-156
- VPCLASS 1-157, 14-19
- Configuring secure connections 17-3
- Connection Manager
 - configuring 10-1
 - failover processing 10-1
 - oncmism 10-1
 - tables
 - syscmism 2-11
 - syscmsmsla 2-12
 - syscmsmtab 2-12
- CONSOLE configuration parameter 1-42
- controlling with onmode -C 14-5
- Conversion messages
 - database server E-49, E-61
 - Enterprise Replication E-61, E-65
- CONVERSION_GUARD configuration parameter 1-42
- Core dump
 - contained in core file A-5
 - DUMPCORE parameter 1-67
- core.pid.cnt file 1-68
- CPU
 - time tabulated 20-202
- CPU virtual processor
 - binding D-2
 - SINGLE_CPU_VP parameter 1-134
- create blobspace argument 21-26
- create chunk argument 21-29
- create database argument 21-31
- create dbspace argument 21-32
- Create Dbpace with On-Monitor 15-3
- CREATE FUNCTION statement 1-159
- CREATE INDEX statement
 - using FILLFACTOR 1-79
- CREATE record subtype (SBLOB) 5-14
- create sbpace argument 21-34
- create sbpace with accesstime argument 21-36
- create sbpace with log argument 21-37
- create tempdbspace argument 21-38
- create tempsbpace argument 21-39
- Creating
 - buffer pool 16-4
 - curlog field 20-217

D

daemon.log A-7

Data distributions

sbspaces 1-148

Data pages

- number to read ahead 1-113
- oncheck -cd 8-8
- oncheck -cD 8-8
- oncheck -cd and -cD 8-3

Data replication

- dr.lostfound file 1-60
- ENCRYPT_CIPHERS 1-71
- ENCRYPT_HDR 1-72
- ENCRYPT_MAC 1-73
- ENCRYPT_MACFILE 1-74
- ENCRYPT_SMX 1-75
- ENCRYPT_SWITCH 1-75

Files

- dr.lostfound 1-60
- flush interval 1-59
- information in sysdri table 2-16
- lost-and-found file 1-60
- wait time for response 1-60

Data row

- big-remainder page 4-14
- forward pointer 4-13
- home page 4-13, 4-14
- locating the row 4-13
- rowid 4-13
- storage strategies 4-12
- storing data on a page 4-14
- TEXT and BYTE data descriptor 4-22

Data-dictionary cache 1-51

Data-distribution cache

- specifying
 - entries 1-64
 - hash buckets 1-61

database server

- monitor status 20-18

Database servers

- blocking 14-4
- bringing online from quiescent mode 14-15, 14-16
- name 1-47
- parallel database query 14-19
- quiescent mode 14-15, 14-16
- remote 2-35
- shutting down 14-15, 14-16
- unblocking 14-4, 21-72

Database tblspace

- entries 4-7
- location in root dbspace 4-1, 4-7
- relation to systable 4-27
- structure and function 4-7
- tblspace number 4-7

Databases

- Database properties 2-14
- effect of creation 4-27
- locale, in sysdbsllocale table 2-15
- owner, in sysmaster database 2-14
- sysdatabases table 2-14

DATASKIP configuration parameter 21-106

- defined 1-43
- using onspaces -f 19-21

DB_LIBRARY_PATH configuration parameter 1-44

DB_LOCALE environment variable 1-138

DBCREATE_PERMISSION configuration parameter 1-44

DBSERVERALIASES configuration parameter 21-74

- defined 1-45
- using onmode -d 14-7

- DBSERVERNAME configuration parameter 21-74
 - defined 1-47
 - using onmode -d 14-7
- dbspaces
 - adding chunk 19-2
 - blobpage structure 4-22
 - creating
 - with onspaces 19-7
 - dropping
 - chunk 19-19
 - with onspaces 19-20
 - ending mirroring 19-24
 - list of structures contained 4-2
 - maximum number 19-5, 19-7, 19-16
 - modifying with onspaces 19-21
 - monitoring with SMI 2-15
 - naming conventions 19-7
 - root name 1-117
 - simple-large-object storage 4-22
 - starting mirroring 19-22
 - storage 4-1
 - structure
 - additional dbspace chunk 4-3
 - chunk free-list page 4-4
 - dbspace 4-1, 4-2
 - mirror chunk 4-3
 - nonroot dbspace 4-2
 - tblspace tblspace 4-4
- DBSPACETEMP configuration parameter 1-48
- DD_HASHMAX configuration parameter 1-50
- DD_HASHSIZE configuration parameter 1-51
- Deadlock 1-51
- DEADLOCK_TIMEOUT configuration parameter 1-51
- Decision-support queries
 - DS_MAX_QUERIES configuration parameter 1-61
 - DS_TOTAL_MEMORY configuration parameter 1-65
 - gate information 20-116
 - gate numbers 20-116
 - MAX_PDQPRIORITY configuration parameter 1-94
 - setting parameters with onmode 14-11
- DEF_TABLE_LOCKMODE configuration parameter 1-52
- Default configuration file 1-2, A-9
- DEFAULTESCCHAR configuration parameter 1-53
- defragment
 - function syntax 21-41
- DELAY_APPLY configuration parameter 1-53
- DELETE
 - logical-log record 5-3
 - record subtype, SBLOB 5-14
- DELITEM logical-log record 5-3
- demonstration databases
 - overview xxiv
 - sales_demo xxiv
 - stores_demo xxiv
 - superstores_demo xxiv
- DERASE logical-log record 5-3
- Descriptor, TEXT and BYTE data 4-22
- DFADDEXT log record 5-3
- DFDRPEXT log record 5-3
- DFEND log record 5-3
- DFMVPG log record 5-3
- DFREMDUM log record 5-3
- DFSTART log record 5-3
- Diagnostic
 - using onmode B-1
- DINDEX logical-log record 5-3
- Direct I/O
 - enabling 1-54
- DIRECT_IO configuration parameter 1-54
- DIRECTIVES configuration parameter 1-55
- Disabilities, visual
 - reading syntax diagrams G-1
- Disability G-1
- DISABLE_B162428_XA_FIX configuration parameter 1-56
- Disabling SQL statement cache 14-13
- Disk I/O
 - buffers 1-38
 - PDQ resources 1-94
- Disk page
 - page compression 4-15
 - storing data on a page 4-14
 - structure
 - blobpage blobpage 4-8
 - dbspace page 4-12
 - types of pages in an extent 4-8, 4-9
- Disk space
 - allocating
 - for system catalogs 4-27
 - when a database is created 4-27
 - when a table is created 4-28
 - chunk free-list page 4-4
 - initializing (oninit -i) 12-1
 - list of structures 4-1
 - maximum chunk size 19-2, 19-5, 19-7, 19-11, 19-16
 - page compression 4-15
 - tracking available space in
 - blobpage 4-23
 - chunk 4-4
- Distributed transactions
 - killing 14-29
- Dotted decimal format of syntax diagrams G-1
- DRAUTO configuration parameter 1-57
- DRDA_COMMBUFFSIZE 1-57
- DRIDXAUTO configuration parameter 1-59
- DRINTERVAL configuration parameter 1-59
- DRLOSTFOUND configuration parameter 1-60
- drop blobpage argument 21-43
- drop chunk argument 21-44
- drop database argument 21-46
- drop dbspace argument 21-46
- DROP DISTRIBUTIONS keyword 1-149
- drop log argument 21-47
- drop sbspace argument 21-49
- drop tempdbspace argument 21-50
- DRPBSP logical-log record 5-3
- DRPCHK logical-log record 5-3
- DRPDBS logical-log record 5-3
- DRPLOG logical-log record 5-3
- DRTIMEOUT configuration parameter 1-60
- DS_HASHSIZE configuration parameter 1-61, 1-64
- DS_MAX_QUERIES configuration parameter
 - changing value 14-11
 - defined 1-61
- DS_MAX_SCANS configuration parameter
 - changing value 14-11
 - defined 1-62
- DS_NONPDQ_QUERY_MEM configuration parameter 1-63
- DS_POOLSIZE configuration parameter 1-64
- DS_TOTAL_MEMORY configuration parameter
 - changing value 14-11
 - defined 1-65
- DUMPCNT configuration parameter 1-67
- DUMPCORE configuration parameter 1-67

- DUMPDIR configuration parameter
 - af.xxx assertion failure file A-4
 - defined 1-67
 - gcore file A-6
 - shmem file A-10
- DUMPGCORE configuration parameter 1-68
- DUMPSHMEM configuration parameter 1-69
 - with onstat -o 20-198
- Dynamic log messages E-65, E-67
- DYNAMIC_LOGS configuration parameter 1-69

E

- EILSEQ_COMPAT_MODE configuration parameter 1-70
- Emergency boot file A-5
- ENABLE_SNAPSHOT_COPY configuration parameter 1-70
- Enabling SQL statement cache 14-13
- ENCRYPT_CIPHERS configuration parameter 1-71
- ENCRYPT_HDR configuration parameter 1-72
- ENCRYPT_MAC configuration parameter 1-73
- ENCRYPT_MACFILE configuration parameter 1-74
- ENCRYPT_SMX configuration parameter 1-75
- ENCRYPT_SWITCH configuration parameter 1-75
- Encrypting or decrypting files 17-1
- Encryption
 - cipher renegotiation 1-75
 - high-availability data replication 1-72
 - MAC files, specifying 1-74
 - message authentication code generation 1-73
 - specifying ciphers and modes 1-71
- ENDTRANS logical-log record 5-3
- Enterprise Replication
 - CDR log record 5-3
 - messages E-61, E-65
 - renaming spaces 19-25
 - SQL administration API functions 21-22
- Environment configuration file A-6
- environment variable
 - setting 1-2
- Environment variables
 - DB_LOCALE 1-138
 - IFX_BATCHEDREAD_TABLE 1-34
 - IFX_DEF_TABLE_LOCKMODE 1-52
 - IFX_DIRECTIVES 1-55, 1-76
 - IFX_XASTDCOMPLIANCE_XAEND 1-56
 - INFORMIXOPCACHE 1-106
 - INFORMIXSERVER 1-47, 14-23
 - onconfig
 - setting 1-2
 - ONCONFIG
 - defined A-9
 - onstat -c 20-29
 - OPTCOMPIND 1-106
 - SERVER_LOCALE 20-86
 - STMT_CACHE 1-142, 14-13
 - viewing Enterprise Replication settings 20-59
- ERASE logical-log record 5-3
- Errors
 - troubleshooting B-1
- Event alarm
 - ALARMPROGRAM parameter 1-27, C-1
 - automatic log backup C-1
 - creating 2-3
 - defined C-1
 - using ex_alarm.sh C-1
 - writing your own script C-1

- Event alarms
 - automatic log backup 1-27
 - class message parameter C-4
 - exit code C-2
 - ID parameter C-4
 - Informix C-5
 - messages C-5
 - ph_alert table C-3
 - RS secondary server C-55
 - SD secondary server C-56
 - see also parameter C-4
 - severity 5 C-50
 - severity codes C-4
 - specific message parameter C-4
- event classes 2-31
- Event severity codes C-4
- ex_alarm.sh script 1-27, C-1
- Exclusive-access, high-water mark 1-91
- EXECUTE FUNCTION statement 21-1
- EXIT_STATUS exit code C-2
- EXPLAIN_STAT configuration parameter 1-76
- EXT_DIRECTIVES configuration parameter 1-76
- EXTDIRECTIVES session environment variable 1-76
- EXTEND record subtype, SBLOB 5-14
- EXTENT_SIZE tag 19-14
- Extents
 - automatic doubling of size 4-11
 - default size 4-8
 - disk page types 4-8, 4-9
 - merging 4-11
 - next-extent, allocating 4-10, 4-11
 - procedure for allocating 4-10
 - size
 - index fragments 4-8
 - initial extent 4-7
 - next extent 4-10
 - structure 4-7
 - sysextents table 2-17
- External backup
 - commands 14-4
- EXTSHMADD configuration parameter 1-77
- Extspace
 - creating 19-16
 - dropping 19-20
 - naming conventions 19-16
 - specifying location 19-16
 - sysextspaces table 2-18

F

- Failover processing 10-1
- FAILOVER_CALLBACK configuration parameter
 - defined 1-78
- FAILOVER_TX_TIMEOUT configuration parameter 1-78
- FAST_RESTART_CKPT_FUZZYLOG configuration parameter D-4
- FAST_RESTART_PHYSLOG configuration parameter D-4
- FASTPOLL configuration parameter
 - defined 1-79
- Files
 - .conf.dbservername A-5
 - .inf.servicename A-7
 - .informix A-6
 - .infos.dbservername A-7
 - .infxdirs A-7
 - .jvpprops A-8
 - ac_config.std A-4

Files *(continued)*

- ac_msg.log A-4
- archecker configuration file A-4
- bar_act.log A-4
- buildsmi_out A-5
- buildsmi.out A-5
- core.pid.cnt 1-68, A-5
- default configuration file A-9
- gcore 1-67, A-6
- informix.rc environment file A-6
- INFORMIXTMP directory A-6
- ISM logs A-7
- ISMVersion A-8
- JVM_vpid A-8
- JVPLOG A-8
- oncfg* A-9
- ONCONFIG A-9
- private environment file A-6
- shmem.pid.cnt 1-69
- shmem.xxx A-10
- sm_versions A-10
- summary of database server files A-1
- VP.servername.nnx A-11
- xbsa.messages A-11
- FILLFACTOR configuration parameter
 - control how indexes fill 4-20
 - defined 1-79
- finderr utility 21-96
- Flushing
 - data-replication buffer 1-59
 - SQL statement cache 14-13
- Force option, onspaces 19-20
- Forced residency
 - starting and ending with onmode 14-18
- Forest of trees indexes 4-16
- Formula
 - quantum of memory 20-116
- Forward pointer
 - blob space blobpage 4-23
 - dbspace storage of simple large objects 4-22
 - defined 4-13
- Fragment
 - index 4-8
 - internal structure of tables 4-15
 - rowids 4-13
 - table, using primary keys 4-14
 - turning DATASKIP ON or OFF for 19-21
 - warning returned when skipped during query 1-43
- Fragment-level statistics 1-148
- FREE_RE logical-log record 5-3
- Free-map page, blob space 4-23
- Freeing
 - blobpages 20-34
 - unused memory segments 14-13
- FULL_DISK_INIT configuration parameter 1-80
- Functional index 4-21
- Functions, SQL administration API
 - add bufferpool argument 21-14
 - add chunk argument 21-15
 - add log argument 21-16
 - add memory argument 21-17
 - add mirror argument 21-18
 - admin() 21-1
 - argument size 21-2
 - return codes 21-2
 - syntax rules 21-1
 - alter chunk argument 21-19

Functions, SQL administration API *(continued)*

- alter logmode argument 11-1, 21-20
- alter plog argument 21-21
- archive fake argument 21-21
- arguments by functional category 21-4
- cdr argument 21-22
- check data argument 21-23
- check extents argument 21-24
- check partition argument 21-24
- checkpoint argument 21-25
- clean sbspace argument 21-26
- compress argument 21-126
- compression purge_dictionary argument 21-132
- create blob space argument 21-26
- create blob space from storagepool argument 21-28
- create chunk argument 21-29
- create chunk from storagepool argument 21-30
- create database argument 21-31
- create db space argument 21-32
- create db space from storagepool argument 21-33
- create sbspace argument 21-34
- create sbspace from storagepool argument 21-35
- create sbspace with accesstime argument 21-36
- create sbspace with log argument 21-37
- create tempdb space argument 21-38
- create tempdb space from storagepool argument 21-39
- create tempsbspace argument 21-39
- create tempsbspace from storagepool argument 21-41
- create_dictionary argument 21-126
- drop blob space argument 21-43
- drop blob space to storagepool argument 21-44
- drop chunk argument 21-44
- drop chunk to storagepool argument 21-45
- drop database argument 21-46
- drop db space argument 21-46
- drop db space to storagepool argument 21-47
- drop log argument 21-47
- drop sbspace argument 21-49
- drop sbspace to storagepool argument 21-49
- drop tempdb space argument 21-50
- drop tempdb space to storagepool argument 21-50
- drop tempsbspace to storagepool argument 21-51
- estimate_compression argument 21-126
 - command output 21-131
- file status 21-51
- for compressing data 21-125, 21-126
- fragment argument 21-126
- ha make primary argument 21-52
- ha rss add argument 21-54
- ha rss argument 21-53
- ha rss change argument 21-55
- ha rss delete argument 21-56
- ha sds clear argument 21-56
- ha sds primary argument 21-57
- ha sds set argument 21-58
- ha set idxauto argument 21-59
- ha set ipl argument 21-59
- ha set primary argument 21-60
- ha set secondary argument 21-61
- ha set standard argument 21-61
- ha set timeout argument 21-62
- message log delete 21-63
- message log rotate 21-64
- message log truncate 21-64
- modify chunk extend argument 21-65
- modify chunk extendable argument 21-66
- modify chunk extendable off argument 21-67

Functions, SQL administration API *(continued)*

- modify space expand argument 21-68
- modify space sp_sizes argument 21-69
- onbar argument 21-92
- onmode and a arguments 21-71
- onmode and c arguments 21-72
- onmode and C arguments 21-73
- onmode and d arguments 21-74
- onmode and D arguments 21-76
- onmode and e arguments 21-76
- onmode and F arguments 21-77
- onmode and j arguments 21-78
- onmode and l arguments 21-79
- onmode and m arguments 21-79
- onmode and M arguments 21-80
- onmode and n arguments 21-81
- onmode and O arguments 21-81
- onmode and p arguments 21-82
- onmode and Q arguments 21-83
- onmode and r arguments 21-84
- onmode and S arguments 21-85
- onmode and W arguments 21-85
- onmode and wf arguments 21-87
- onmode and wm arguments 21-88
- onmode and Y arguments 21-90
- onmode and z arguments 21-91
- onmode and Z arguments 21-91
- onmode, wm, and AUTO_LRU_TUNING arguments 21-89
- onsmsync argument 21-93
- ontape archive argument 21-95
- print error argument 21-96
- print file info argument 21-96
- print partition argument 21-97
- rename space argument 21-98
- repack argument 21-126
- repack_offline argument 21-126
- reset sysadmin argument 21-99
- restart listen argument 21-100
- scheduler argument 21-101
- scheduler lmm disable argument 21-104
- scheduler lmm enable argument 21-101
- set chunk argument 21-105
- set dataskip argument 21-106
- set index compression argument 21-106
- set onconfig memory argument 21-107
- set onconfig permanent argument 21-108
- set sbospace accesstime argument 21-109
- set sbospace avg_lo_size argument 21-110
- set sbospace logging argument 21-111
- set sql tracing argument 21-111
- set sql tracing database argument 21-113
- set sql tracing session argument 21-114
- set sql tracing user argument 21-115
- set sql user tracing argument 21-115
- shrink argument 21-126
- start listen argument 21-116
- start mirroring argument 21-117
- stop listen argument 21-117
- stop mirroring argument 21-118
- storagepool add argument 21-119
- storagepool delete argument 21-121
- storagepool modify argument 21-122
- storagepool purge argument 21-124
- table argument 21-126
- task() 21-1
 - argument size 21-2

Functions, SQL administration API *(continued)*

- task() *(continued)*
 - return codes 21-2
 - syntax rules 21-1
- uncompress argument 21-126
- uncompress_offline argument 21-126

G

- Gateway transactions 1-81
- gcore
 - file A-6
 - utility 1-67, 1-68
- genoncfg utility 7-1
- Global Security Kit 1-80
- Global transactions
 - using onstat -G 20-187
 - using onstat -x 20-217
- GLS-locale database property 2-14
- GSKCapiCmd 1-80
- GSKCmd 1-80
- GSKikm 1-80

H

- ha make primary argument 21-52
- ha rss add argument 21-54
- ha rss argument 21-53
- ha rss change argument 21-55
- ha rss delete argument 21-56
- ha sds clear argument 21-56
- ha sds primary argument 21-57
- ha sds set argument 21-58
- ha set idxauto argument 21-59
- ha set ipl argument 21-59
- ha set primary argument 21-60
- ha set secondary argument 21-61
- ha set standard argument 21-61
- ha set timeout argument 21-62
- HA_ALIAS configuration parameter
 - defined 1-81
- Hash buckets
 - data-dictionary cache 1-51
 - data-distribution cache
 - specifying hash buckets 1-61
- HDELETE logical-log record 5-3
- HDRUPD record subtype, SBLOB 5-14
- HETERO_COMMIT configuration parameter 1-81
- Heterogeneous-commit transactions 1-81
- HEURTX logical-log record 5-3
- High-water mark, transaction 1-92
- HINSERT logical-log record 5-3
- Home page 4-13, 4-14
- HUPAFT logical-log record 5-3
- HUPBEF logical-log record 5-3
- HUPDATE logical-log record 5-3

I

- I/O
 - lightweight 19-13
- IBM Informix Server Administrator
 - adding or dropping logs 1-87
 - defined 6-2
 - setting onconfig parameters 1-2
- IBM Informix STAR queries 20-215

IBM Informix Storage Manager

- catalog A-7
- ISMversion file A-8
- logs A-7
- sm_versions file A-10

Identifier, defined 1-47

IDXFLAGS logical-log record 5-3

ifx_allow_newline() routine 1-28

IFX_BATCHEDREAD_TABLE environment variable 1-34

IFX_DEF_TABLE_LOCKMODE environment variable 1-52

IFX_DIRECTIVES environment variable 1-55, 1-76

IFX_EXTEND_ROLE configuration parameter 1-82

IFX_FOLDVIEW configuration parameter 1-82

ifx_lo_specset_estbytes function 19-14, 19-15

ifx_lo_stat function 19-18

IFX_XASTDCOMPLIANCE_XAEND environment variable 1-56

ifxclone utility 18-1

ifxcollect tool B-2

IKEYCMD 1-80

iKeyman 1-80

illsrta.xx file A-6

Index

- branch node 4-16
- configuration 4-20
- duplicate key values 4-19
- forest of trees 4-16
- functional 4-21
- how created and filled 4-17
- key value locking 4-20
- leaf node 4-16
- reuse of freed pages 4-20
- root node 4-16
- structure of B-tree 4-16

Index item

- calculating the length of 4-20

Index page

- compactness 1-79
- creation of first 4-17
- effect of creation 4-17
- structure 4-16

indexes

- defragmenting 21-41

industry standards xxv

Information

- SQL profile 2-40

informix.rc environment file A-6

INFORMIXOPCACHE environment variable 1-106

INFORMIXSERVER environment variable 1-45, 1-47, 14-23

INFORMIXTMP directory A-6, A-7

Initializing

- disk structures 4-1

INSERT logical-log record 5-3

InstallServer.log file A-7

Interrupt signal 6-1

Interval

- checkpoint 1-41

ISA. 1-87

ism_startup command A-10

ISM.

- See IBM Informix Storage Manager. A-7

ISMversion file A-8

ISOSPCOMMIT log record 5-3

J

JDKVERSION discontinued configuration parameter D-5

JDKVERSION parameter D-5

JVM_vpid file A-8

JVPDEBUG configuration parameter A-8

JVPLOG file A-8

JVPLOGFILE configuration parameter A-8

K

Key value

- checking order with oncheck 8-3, 8-11

- duplicates 4-19

- locking 4-20

Key-only

- inserting entries 1-144, 14-24

Killing a session 14-29

L

Large chunk mode 14-3

Latch

- displaying with onstat -s 20-22, 20-209

- identifying the resource controlled 20-209

LBU_PRESERVE discontinued configuration parameter D-5

LCKLVL logical-log record 5-3

LG_ADDBPOOL logical-log record 5-3

LG_CDINDEX log record 5-3

Licensed users, maximum allowed E-25

Lightweight I/O 19-13

LIMITNUMSESSIONS configuration parameter 1-83

Limits

- SQL statement cache size 1-144, 14-24

- virtual processors 14-19

Linking, name of root dbspace 1-118

Listen threads

- starting dynamically 14-22, 21-116

- stopping and restarting dynamically 14-22, 21-100

- stopping dynamically 14-22, 21-117

LISTEN_TIMEOUT configuration parameter 1-84

Listener thread 1-45

Locales xxiv

Location, extspace 19-16

Lock

- buffer-access-level flag bits 20-27

- information in syslocks table 2-21

- key-value 4-20

- maximum time to acquire 1-51

- monitoring with onstat -k 20-21, 20-192

- multiprocessor 1-97

- oncheck options 8-2

- type codes for onstat -k 20-192

Lock mode, page or row 1-52

LOCK_MODE tag 19-14

LOCKS configuration parameter 1-84

Log File Required event alarm 21-16

Log position 20-215, 20-217

log_full scripts 1-27, C-1

LOG_INDEX_BUILDS configuration parameter 1-87, 21-59

LOG_STAGING_DIR configuration parameter 1-87

LOGBUFF configuration parameter 1-85

LOGFILES configuration parameter 1-86

Logging

- blobspace free-map page 4-23

- flags for mode 4-7

LOGGING tag 19-14

- Logical character semantics 1-138
- Logical log
 - adding files 16-2
 - backup
 - alarm triggered 1-27, C-1
 - dropping files 16-2
 - file
 - created during initialization 1-86
 - displaying contents 13-1
 - log position 20-217
 - moving 16-2
 - reading the log file 13-1
 - size 1-88
 - switching with onmode 14-18
 - files
 - maximum number 16-2
 - minimum number 16-2
 - in root dbspace 4-1
 - log position 20-215, 20-217
 - maximum size 20-215
 - monitoring with SMI 2-22
 - onparams
 - adding files 16-2
 - dropping files 16-2
 - record
 - additional columns 5-3
 - checkpoint 5-2
 - displaying 13-1
 - distributed transactions 5-2
 - DROP TABLE operation 5-1
 - generated by rollback 5-1
 - header columns 5-2
 - types 5-3, 5-14
 - specifying file size 16-2
- Logical page contents
 - displaying with oncheck 8-16
- Logical recovery, number of threads 1-104
- Logical-log buffer and LOGBUFF configuration parameter 1-85
- logmessage table E-1
- logposit field 20-217
- LOGSIZE configuration parameter 1-88
- LOGSMAX discontinued configuration parameter D-5
- Long transaction
 - high-water mark 1-92
 - LTXEHWM 1-91
 - LTXHWM 1-92
- Loosely-coupled mode 20-217
- LOW_MEMORY_MGR configuration parameter 1-89
- LOW_MEMORY_RESERVE configuration parameter 1-90
- LRU
 - changing dynamically 14-27
- LRU queues
 - displaying with onstat -R 20-22, 20-207
 - FLRU queues 20-207
 - MLRU queues 20-207
 - modified pages, percentage 1-39
- LRU_MAX_DIRTY discontinued configuration parameter D-6
- lru_max_dirty field in the BUFFERPOOL configuration parameter 1-39
- LRU_MIN_DIRTY discontinued configuration parameter D-6
- lru_min_dirty field in the BUFFERPOOL configuration parameter 1-39
- LRUS discontinued configuration parameter D-7
- lrus field in the BUFFERPOOL configuration parameter 1-38
- LTXEHWM configuration parameter 1-91
- LTXHWM configuration parameter 1-92

M

- MAX_FILL_DATA_PAGES configuration parameter 1-92
- MAX_INCOMPLETE_CONNECTIONS configuration parameter 1-93
- MAX_PDQPRIORITY configuration parameter
 - changing value 14-11
 - defined 1-94
- MaxConnect
 - DBSERVERALIASES configuration parameter 1-45
 - DBSERVERNAME configuration parameter 1-47
 - NETTYPE configuration parameter 1-98
- Maximum number
 - chunks 19-2
 - storage spaces 19-5, 19-7, 19-11, 19-16
- Maximum user connections E-25
- Memory
 - freeing unused segments 14-13
 - pools, SQL statement cache 1-145
 - quantum allocated by MGM 20-116
 - specifying size of 1-106
- Memory Grant Manager
 - monitoring resources 20-116
- Memory information
 - SMI tables 2-24, 2-34
- Message authentication code files 1-73, 1-74
- Message log
 - alphabetical listing of messages E-1
 - categories of messages E-2
 - defined A-8, E-1
 - displaying with onstat -m 20-21, 20-197
 - event alarms C-2
 - location 1-96
 - viewing messages E-1
- Messages
 - A-B E-2, E-4
 - assertion-failed E-2
 - C E-4, E-14
 - changing sbspace minimum extent size 19-14
 - conversion and reversion E-49, E-61
 - D-E-F E-14, E-19
 - dynamic log E-65, E-67
 - Enterprise Replication E-61, E-65
 - G-H-I E-19, E-22
 - in-place ALTER TABLE E-18
 - J-K-L-M E-22, E-27
 - N-O-P E-27, E-36
 - onspaces E-33
 - Q-R-S E-36, E-41
 - sbspace metadata E-67, E-68
 - symbols E-48, E-49
 - T-U-V E-41, E-47
 - truncate table E-68
 - turning off smart-large-object logging 19-14
 - W-X-Y-Z E-47, E-48
- Metadata
 - area, structure 4-25
 - checking with oncheck 8-1
 - creating 19-11
 - messages E-67, E-68
 - size 19-11, 19-13
 - specifying offset 4-24, 19-11
 - specifying size 19-11
 - temporary sbspace 1-123
- MGM
 - information 2-23
- mi_lo_decrefcount() function 19-18
- mi_lo_increfcount() function 19-18

- mi_lo_specset_estbytes() function 19-14, 19-15
- mi_lo_stat function() 19-18
- Microsoft Transaction Server
 - defined 20-215
 - onstat -x output 20-217
- MIN_EXT_SIZE tag 19-14
- Mirror chunk, structure 4-3
- MIRROR configuration parameter 1-95
- Mirroring
 - changing chunk status 19-26
 - enable flag 1-95
 - initial chunk 1-95
 - starting 19-22
 - stopping 19-24
- MIRROROFFSET configuration parameter 1-95
- MIRRORPATH configuration parameter 1-95
- Modes
 - encryption 1-71
- Modified pages
 - specifying percentage
 - LRU queue 16-4
- monitor server status
 - onstat utility 20-18
- Monitoring
 - display environment variables 20-86
 - distributed queries 20-215
 - licensed user connections E-25
 - MGM resources 20-116
- Moving logical-log files 16-2
- MSG_DATE configuration parameter 1-96
- MSGPATH configuration parameter 1-96
- Multiprocessor computer
 - AFF_SPROC configuration parameter D-3
 - processor affinity 1-161
- MULTIPROCESSOR configuration parameter 1-97
- Mutex threads 20-154
- MVIDXND logical-log record 5-3

N

- Name
 - blob space 19-5
 - db space 19-7
 - ext space 19-16
 - sbspaces 19-11
- Name service cache 1-101
- NET_IO_TIMEOUT_ALARM 1-97
- NETTYPE configuration parameter
 - defined 1-98
 - tuning example 1-98
- Network information
 - SMI tables 2-24, 2-25
- Newline character, quoted strings 1-28
- NEXT_SIZE tag 19-15
- Next-extent
 - allocation 4-10
 - allocation strategy 4-11
 - doubling of size 4-11
 - initial size 4-7
 - nonfragmented table 4-8
- no_log scripts 1-27
- NOAGE discontinued configuration parameter D-7
- Node, index
 - branch 4-16
 - creating 4-18
 - what points to 4-18
- checking horizontal and vertical nodes 8-3, 8-11

- Node, index (*continued*)
 - defined 4-16
 - leaf 4-16
 - contents 4-18
 - pointer 4-18
 - root node 4-16
 - creating 4-17
 - when fills 4-18
 - types 4-16
- Non-default page size
 - physical log 16-4
- NS_CACHE configuration parameter 1-101
- NSF lock contention 1-102
- NUMAIOVPS configuration parameter
 - defined D-8
- Number of page-cleaner threads 1-41
- NUMCPUVPS configuration parameter
 - defined D-9
- NUMFDSERVERS configuration parameter 1-102

O

- OFF_RECVRY_THREADS configuration parameter 1-103
- Offset
 - mirrored chunk 19-11
 - size 19-2, 19-5, 19-7, 19-11, 19-16
- omode -wf or -wm
 - changing configuration parameters 14-25
 - usage 14-25
- ON_RECVRY_THREADS configuration parameter 1-104
- ON-Bar utility
 - activity log A-4
 - emergency boot files A-5
 - sm_versions file A-10
 - system tables 2-6
 - xbsa.messages log A-11
- On-Monitor Create Dbsapce name 15-3
- ON-Monitor utility
 - Archive menu options 15-2
 - changing
 - database server mode 14-17
 - Dbspaces menu options 15-2
 - Diagnostics screen 15-3
 - displaying system page size 1-40
 - executing shell commands 15-2
 - Force-Ckpt menu options 15-2
 - help 15-1
 - Initialization screen 15-3
 - Logical-Logs menu options 15-2
 - Mode menu options 15-2
 - navigating 15-1
 - Parameters menu options 15-2
 - Performance screen 15-3
 - Shared-memory screen 15-3
 - Status menu options 15-2
 - using 15-1
- onbar argument 21-92
- oncfg file
 - and onspaces 19-1
 - defined A-9
- oncheck -pt command 21-97
- oncheck -pT command 21-97
- oncheck utility
 - check-and-repair 8-1
 - defined 8-1
 - display reserved, physical-log, and logical-log pages 4-2
 - list of functions 8-1

- oncheck utility *(continued)*
 - locking 8-3
 - oncheck utility
 - space required for sorting 8-1
 - option descriptions 8-3
 - options
 - cc 8-8
 - cd 8-8
 - cD 8-8
 - ce 8-3, 8-10
 - ci 8-11
 - cl 8-11
 - cr 8-12
 - cR 8-12
 - cs 8-13
 - cS 8-13
 - n 8-2, 8-3
 - pB 8-13
 - pc 8-8
 - pd 8-13
 - pD 8-13
 - pe 4-1, 8-10
 - pk 8-15
 - pK 8-15
 - pl 8-15
 - pL 8-15
 - pp 8-16
 - pP 8-16
 - pr 1-2, 4-2, 8-18
 - pR 4-2, 8-18
 - ps 8-13
 - pS 8-13
 - pt 8-19
 - pT 8-19
 - u 8-22
 - x 8-21
 - y 8-2, 8-3
 - overview of functionality 8-1
 - suppressing messages 8-3
 - syntax 8-3
- onclean utility 9-1
- oncmism utility
 - managing high-availability servers 10-1
- onconfig configuration file
 - displaying 1-2
 - format 1-1
 - preparing 1-2
 - templates 1-2
- ONCONFIG configuration file
 - defined A-9
 - displaying 20-21, 20-29
 - setting with genoncfg utility 7-1
 - using onstat -c 20-29
- ONCONFIG environment variable
 - ONCONFIG file A-9
- ONCONFIG parameters
 - LOG_INDEX_BUILDS 21-59
- onconfig.std template file
 - defined 1-2
 - printing 1-2
- ondblog utility 11-1
 - BAR_ACT_LOG file 11-1
 - defined 11-1
- ONDBSPACEDOWN configuration parameter 21-81
 - defined 1-104
 - overriding WAIT mode 14-19
- oninit
 - i option
 - affect of FULL_DISK_INIT 1-80
 - return codes 12-4
- oninit utility
 - option descriptions 12-1
 - options
 - i 12-1
 - j 12-1
 - s 12-1
 - starting database server 12-1
- ONLIDX_MAXMEM configuration parameter 1-105
- onlog utility
 - defined 13-1
 - filters for logical-log records
 - displaying 13-3
 - reading 5-3, 13-1
 - options
 - b 13-1
 - c 13-4
 - d 13-1
 - l 5-3, 13-3
 - n 13-1
 - q 13-1
 - t 13-4
 - u 13-4
 - x 13-4
- onmode -c command 21-72
- onmode -D command 21-76
- onmode -e command 21-76
- onmode -F command 21-77
- onmode -j command 21-78
- onmode -l command 21-79
- onmode -m command 21-79
- onmode -M command 21-80
- onmode -n command 21-81
- onmode -O command 21-81
- onmode -p command 21-82
- onmode -Q command 21-83
- onmode -r command 21-84
- onmode -S command 21-85
- onmode -W command 21-85
- onmode -wf command 21-87, 21-108
- onmode -wm AUTO_LRU_TUNING command 21-89
- onmode -wm command 21-88, 21-107
- onmode -Y command 21-90
- onmode -z command 21-91
- onmode -Z command 21-91
- onmode and a arguments 21-71
- onmode and c arguments 21-72
- onmode and C arguments 21-73
- onmode and d arguments 21-74
- onmode and D arguments 21-76
- onmode and e arguments 21-76
- onmode and F arguments 21-77
- onmode and j arguments 21-78
- onmode and l arguments 21-79
- onmode and m arguments 21-79
- onmode and M arguments 21-80
- onmode and n arguments 21-81
- onmode and O arguments 21-81
- onmode and p arguments 21-82
- onmode and Q arguments 21-83
- onmode and r arguments 21-84
- onmode and S arguments 21-85
- onmode and W arguments 21-85
- onmode and wf arguments 21-87

- onmode and wm arguments 21-88
- onmode and Y arguments 21-90
- onmode and z arguments 21-91
- onmode and Z arguments 21-91
- onmode syntax 14-1
- onmode utility
 - P option 14-22
 - adding
 - shared-memory segment 14-2
 - virtual processors 14-19
 - administration mode 14-17
 - blocking the database server 14-4
 - caching the allowed.surrogates file 14-6
 - changing
 - database server mode 14-15, 14-17
 - DS_MAX_QUERIES 14-11
 - DS_MAX_SCANS 14-11
 - DS_TOTAL_MEMORY 14-11
 - MAX_PDQPRIORITY 14-11
 - shared-memory residency 14-18
 - SQL statement cache usage 14-13, 14-24
 - changing SQL statement cache 1-142
 - defined 14-1
 - forcing a checkpoint 14-4
 - freeing memory segments 14-13
 - killing
 - distributed transactions 14-29
 - session 14-29
 - marking disabled dbspace as down 14-19
 - options
 - a 14-2
 - BC 1 14-3
 - BC 2 14-3
 - c block 14-4
 - c unblock 14-4
 - cache surrogates 14-6
 - d 14-7
 - D 14-11
 - e 1-142, 14-13
 - F 14-13
 - I 14-14
 - j 14-15
 - j -U 14-17
 - k 14-15, 14-16
 - l 14-18
 - m 14-15, 14-16
 - M 14-11
 - n 14-18
 - O 14-19
 - p 14-19
 - Q 14-11
 - r 14-18
 - R 14-23
 - s 14-15, 14-16
 - S 14-11
 - u 14-15, 14-16
 - W 14-24
 - Y 14-28
 - y confirm action 14-1
 - z 14-29
 - Z 14-29
 - PDQ 20-116
 - regenerating .infos file 14-23
 - removing virtual processors 14-19
 - rereading the allowed.surrogates file 14-6
 - setting
 - data replication type 14-7
- onmode utility (*continued*)
 - setting (*continued*)
 - decision-support parameters 14-11
 - Starting or ending forced residency 14-18
 - switching logical-log files 14-18
 - trapping errors 14-14, B-1
 - unblocking the database server 14-4
- onmode, wm, and AUTO_LRU_TUNING arguments 21-89
- onparams utility 16-1
 - adding logical-log file 16-2
 - backing up changes to physical log 16-4
 - changing physical log size and location 16-3
 - defined 16-1
 - dropping a logical-log file 16-2
 - examples 16-6
- onpassword utility
 - configuring secure connections 17-3
 - encrypting or decrypting text files 17-1
- onpload
 - onstat -j utility 20-190
- onshutdown script 9-2
- onsmsync argument 21-93
- onsnmp log file A-10
- onsocimc protocol 1-98
- onspaces
 - syntax 19-1
- onspaces -f command 21-106
- onspaces -m command 21-117
- onspaces -r command 21-118
- onspaces -ren command 21-98
- onspaces utility
 - Df options 19-12
 - adding a chunk to
 - dbspace or blobspace 19-2
 - sbspaces 19-3
 - avoid chunk overwrite 19-27
 - changing chunk status 19-26
 - changing sbspace defaults 19-15, 19-17
 - cleaning up sbspaces 19-18
 - creating a blobspace 19-5
 - creating a temporary sbspace 19-11
 - creating an extspace 19-16
 - creating an sbspace 19-11
 - defined 19-1
 - dropping a blobspace, dbspace, extspace, or sbspace 19-20
 - dropping a chunk 19-19
 - ending mirroring 19-24
 - forcing a drop 19-20
 - options
 - a 19-2, 19-3
 - b 19-5
 - c 19-5, 19-7, 19-16
 - cl 19-18
 - d 19-19, 19-20
 - Df 19-12
 - f 19-21
 - g 19-5, 19-16
 - l 19-16
 - m 19-22
 - Mo 19-3, 19-11
 - Ms 19-3, 19-11
 - r 19-24
 - s 19-26
 - S 19-11
 - t 19-7, 19-10, 19-12
 - x 19-16
 - specifying DATASKIP 19-21

onspaces utility *(continued)*
starting mirroring 19-22

onsrvapd.log file A-10

onstat -g options 20-42

onstat utility

- option 20-23
- option 20-19, 20-25
- a option 20-20, 20-26
- b option 1-40, 20-21, 20-26
- B option 20-21, 20-25, 20-27
- c option 20-21, 20-29
- C option 20-21, 20-29
- d option 1-48, 20-21, 20-34
- D option 20-21, 20-39
- d update option 20-34
- f option 1-43, 20-21
- F option 1-41, 20-21, 20-40
- g afr option 20-48
- g aqt option 20-49
- g ath option 20-25
- g cdr option 20-58
- g ckp option 20-61
- g cluster option 20-65
- g con option 20-148
- g cpu option 20-71
- g dbc option 20-73
- g ddr option 20-74
- g defragment option 20-76
- g dic option 20-76
- g dis option 20-77
- g dll option 20-79
- g dmp option 20-80
- g dri ckpt option 20-81
- g dri option 20-81
- g dsc option 20-83
- g dsk option 20-79
- g dss command 20-84
- g dtc command 20-85
- g env option 20-86
- g ffr option 20-88
- g glo option 20-89
- g glo options A-8
- g grp command 20-91
- g his option 20-97
- g ioa option 20-102
- g iob option 20-103
- g iof option 20-104
- g iog option 20-105
- g ioq option 20-105
- g iov option 20-107
- g ipl option 20-107
- g lap option 20-109
- g laq option 20-109
- g lmx option 20-113
- g lsc option 20-114
- g mem option 20-115
- g mgm option 14-11, 20-116
- g nbm option 20-119
- g nif option 20-120
- g nsc option 1-98, 20-121
- g nsd option 20-124
- g nss option 20-124
- g ntd option 20-125
- g ntm option 20-126
- g ntt option 20-127
- g ntu option 20-127
- g opn option 20-127

onstat utility *(continued)*

- g option 20-21
- G option 20-21, 20-187
- g options 20-42
- g osi option 20-130
- g pos option 20-130
- g ppd option 20-131
- g ppf option 1-149, 20-132
- g pqs option 20-133
- g prc option 20-134
- g proxy option 20-135
- g qst option 20-141
- g que option 20-140
- g rbm option 20-144
- g rcv option 20-144
- g rea option 20-147
- g rep option 20-148
- g rss option 20-151
- g rwm option 20-154
- g sch option 20-155
- g scn option 20-155
- g sds option 20-157
- g seg option 1-128, 20-25, 20-161
- g ses option 20-162
- g sle option 20-167
- g smb option 20-167
- g smx option 20-169
- g spi option 20-171
- g sql option 20-172
- g src option 20-173
- g ssc all option 20-174
- g ssc option 1-144, 20-174
- g ssc pool option 1-145, 20-174
- g stk option 20-176
- g stm option 20-176
- g stq option 20-177
- g sts option 20-177
- g sym option 20-178
- g sync option 20-179
- g tpf option 20-180
- g ufr option 20-181
- g vpcache option 20-182
- g wai option 20-184
- g wmx option 20-185
- g wst option 20-185
- h option 20-21, 20-188
- i option 20-19, 20-21, 20-25, 20-189
- j option 20-190
- k option 1-84, 20-21, 20-192
- l option 1-86, 16-2, 20-21, 20-193
- L option 20-196
- m option 1-97, 20-21, 20-197, E-1
- o nobuff option 20-25
- o option 20-21, 20-25, 20-198
- O option 1-106, 20-21, 20-198
- options source_file 20-25
- p option 1-52, 20-200
- P option 20-22, 20-25, 20-203
- pu option 20-19
- r option 20-19, 20-22, 20-204
- R option 20-22, 20-207
- s option 20-22, 20-209
- t option 20-22, 20-211
- T option 20-211
- u option 20-22, 20-212
- x option 14-29, 20-22, 20-215
- X option 20-22, 20-217

- onstat utility (*continued*)
 - z option 20-22, 20-219
 - defined 20-1
 - displaying
 - chunk information 20-34
 - ONCONFIG file 20-21
 - freeing blobpages 20-34
 - header 20-23
 - monitor server status 20-18
 - monitoring
 - PDQ 20-116
 - no options 20-23
 - onstat -g lmm 20-111
 - onstat -g rah 20-142
 - print file info about B-tree scanner subsystem and thread 20-29
 - repeated execution
 - r option 20-22
 - seconds parameter 20-19
 - return codes on exit 20-219
 - syntax 20-19
 - table of options 20-19
 - terminating interactive mode 20-189
 - terminating repeating sequence 20-189
 - using SMI tables for onstat information 2-45
- ontape archive argument 21-95
- ontliimc protocol 1-98
- OPCACHEMAX configuration parameter 1-106
- OPT_GOAL configuration parameter 1-106
- OPTCOMPIND
 - configuration parameter 1-106
 - environment variables 1-106
- Optical storage and STAGEBLOB configuration parameter 1-142
- Optical Subsystem memory cache 20-21, 20-198
- Optimizer directives 1-76
- Optimizing hash and nested-loop joins 1-106

P

Page

- bitmap page 4-23
- blobspace blobpage 4-23
- blobspace free-map page 4-23
- components of dbspace page 4-12
- compression 4-15
- dbspace blobpage 4-22
- dbspace page types 4-8, 4-9
- definition of full page 4-14
- free page, defined 4-8, 4-9
- page types in extent 4-8, 4-9
- reuse of index page 4-20
- size, shown with onstat -b 20-26
- structure and storage of 4-12
- Page compression 4-15
- Page header, length of 4-5
- Page zero 1-80
- PAGE_CONFIG reserved page 1-2, 4-2, 8-12
- Page-cleaner threads
 - codes for activity state 20-40
 - monitoring activity 20-21, 20-40
 - numbers 1-41
- Parallel database queries
 - gate information 20-116
 - MGM resources 20-116
 - monitoring resources 20-116
 - monitoring resources allocated 20-116

- partitions
 - defragmenting 21-41
- Partnum field in systables 4-5
- Passwords, encrypted, not shown in onstat -g sql 20-172
- Pathname, specifying 19-5, 19-7, 19-12
- PBDELETE logical-log record 5-3
- PBINSERT logical-log record 5-3
- PC_HASHSIZE configuration parameter 1-108
- PC_POOLSIZE configuration parameter 1-109
- PDELETE record subtype (SBLOB) 5-14
- PDINDEX logical-log record 5-3
- PDQ
 - CPU VPs 14-19
 - DS_MAX_QUERIES configuration parameter 1-61
 - DS_MAX_SCANS configuration parameter 1-62
 - DS_TOTAL_MEMORY configuration parameter 1-65
 - information 2-23
 - MAX_PDQPRIORITY configuration parameter 1-94
 - PDQPRIORITY configuration parameter 1-94
- PDQPRIORITY configuration parameter 1-94
- Pending transaction 20-215
- PERASE logical-log record 5-3
- Performance advisory messages 20-61
- PGALTER logical-log record 5-3
- PGMODE logical-log record 5-3
- ph_alert table 3-5
 - event alarms C-3
- ph_group table 3-5
- ph_run table 3-4
- ph_task table 3-2
- ph_threshold table 3-8
- PHYSBUFF configuration parameter 1-109
- PHYSDBS configuration parameter D-9
- PHYSFILE configuration parameter 1-110
- Physical log
 - backing up
 - changes 16-4
 - files 16-4
 - changing
 - size and location 16-3
 - changing size 16-3
 - root dbspace 4-1
 - size 1-110
 - using non-default page size 16-4
- Physical-log buffer
 - dbspace location D-9
 - size 1-109
- PLOG_OVERFLOW_PATH configuration parameter 1-111
- PN_STAGEBLOB_THRESHOLD configuration parameter 1-111
- PNGPALIGN8 log record 5-3
- PNLOCKID logical-log record 5-3
- PNSIZES logical-log record 5-3
- Pools
 - SQL statement cache 20-174
- PRELOAD_DLL_FILE configuration parameter 1-112
- PREPARE logical-log record 5-3
- Preventing long transactions 1-92
- Primary key, use in fragmented table 4-14
- print error argument 21-96
- print partition argument 21-97
- Printing
 - diagnostics, onmode -I B-1
 - global transactions 20-187
 - onconfig.std template file 1-2
- Priority aging, of CPU virtual processors D-7
- Private environment file A-6

- Processor affinity
 - AFF_NPROCS configuration parameter D-2
 - AFF_SPROC configuration parameter D-3
 - multiprocessors 1-97
 - set with VPCLASS configuration parameter 1-161
- Processor, locking for multiple or single 1-97
- Profile
 - displaying count, onstat -p 20-22, 20-200
 - monitoring with SMI 2-25, 2-26
 - setting counts to zero 20-22, 20-219
- Profile, partition 1-149
- PTADDESC logical-log record 5-3
- PTALTER logical-log record 5-3
- PTALTNEWKEYD log record 5-3
- PTALTOLDKEYD log record 5-3
- PTCOLUMN log record 5-3
- PTEXTEND logical-log record 5-3
- PTRENAME log record 5-3
- PTRUNC record subtype 5-14
- PTRUNCATE log record 5-3

Q

- QSTATS configuration parameter 1-112
 - with onstat-g qst 20-141
- Quantum, of memory 20-116
- Quiescent mode 14-15, 14-16

R

- RA_PAGES configuration parameter 1-113
- Raw disk space
 - UNIX 19-2
 - Windows 19-2, 19-5, 19-7
- RDELETE logical-log record 5-3
- Read-ahead, data pages
 - number 1-113
- Receive Manager in Enterprise Replication 20-144
- Recovery threads
 - offline 1-103
 - online 1-104
- REFCOUNT record subtype 5-14
- Reliable Queue Manager (RQM) in Enterprise Replication 20-148
- Remainder page, defined 4-14
- REMOTE_SERVER_CFG configuration parameter 1-113
- REMOTE_USERS_CFG configuration parameter 1-114
- Removing
 - CPU virtual processors 14-21
 - stray smart large objects 19-18
 - virtual processors 14-19
- rename space argument 21-98
- RENDBS logical-log record 5-3
- REPEVT_CLUST_CHG
 - event class 2-31
 - sub-events 2-31
- REPEVT_CLUST_LATSTAT
 - event class 2-31
 - sub-events 2-31
- REPEVT_CLUST_PERFSTAT
 - event class 2-31
 - sub-events 2-31
- REPEVT_CM_ADM
 - event class 2-31
 - sub-events 2-31

- REPEVT_ER_ADM
 - event class 2-31
 - sub-events 2-31
- REPEVT_SRV_ADM
 - event class 2-31
 - sub-events 2-31
- Reserved area, sbospace 4-24
- Reserved pages
 - checking with oncheck 8-3, 8-12
 - defined 4-2
 - location in root dbspace 4-1
 - viewing contents 4-2
- reset sysadmin argument 21-99
- RESIDENT configuration parameter
 - defined 1-115
- Resident shared memory
 - Configuration parameters
 - RESIDENT 14-18
 - RESIDENT configuration parameter 1-115
 - onmode -r or -n 14-18
 - turning on and off residency 14-18
- restart listen argument 21-100
- RESTARTABLE_RESTORE configuration parameter 1-116
- RESTORE_POINT_DIR configuration parameter 1-117
- Results tables 3-8
- Return codes
 - onstat utility, on exit 20-219
- Reuse of freed index pages 4-20
- revcdr.out file E-61
- revcdr.sh script A-1, A-10, E-61
- Reversion messages
 - database server E-49, E-61
 - Enterprise Replication E-61, E-65
- REVERT logical-log record 5-3
- revtestcdr.out file E-61
- RINSERT logical-log record 5-3
- ROLLBACK logical-log record 5-3
- Rolling back long transactions 1-91
- ROLWORK logical-log record 5-3
- Root dbspace
 - initial chunk 1-118
 - mirroring 1-95
 - specifying ROOTNAME configuration parameter 1-117
 - structure 4-1
 - using a link 1-118
- ROOTNAME configuration parameter
 - defined 1-117
 - use by PHYSDBS D-9
- ROOTOFFSET configuration parameter 1-118
- ROOTPATH configuration parameter
 - defined 1-118
 - specifying as a link 1-95, 1-118
- ROOTSIZE configuration parameter 1-119
- Round-robin fragmentation
 - constraints 1-111
- ROWID
 - defined 4-13
 - fragmented table 4-13
 - functions as forward pointer 4-13
 - locking information 20-192
 - stored in index pages 4-13
- Rows
 - data, storage 4-14
 - displaying contents with oncheck 8-13
 - storage location 4-14
- RS secondary server
 - event alarms and messages C-55

- RSS_FLOW_CONTROL configuration parameter 1-119
- RSVEXTEN logical-log record 5-3
- RTO_SERVER_RESTART configuration parameter 1-120
- RTREE logical-log record 5-3

S

- sales_demo
 - demonstration database xxiv
- sample databases
 - See* demonstration databases
- SBLOB logical-log record 5-14
- Sbpage structure 4-26
- SBSPACENAME configuration parameter 1-122, 1-148
- sbspaces
 - g option 19-16
 - adding a chunk 19-3
 - changing defaults 19-15, 19-17
 - cleaning up references 19-18
 - creating with onspaces 19-11
 - default name 1-122
 - dropping a chunk 19-19
 - dropping a sbspace 19-20
 - ending mirroring 19-24
 - maximum number 19-5, 19-7, 19-11, 19-16
 - metadata area
 - size and offset 19-11, 19-13
 - structure 4-25
 - multiple chunk 4-26
 - naming conventions 19-11
 - onstat -d usage 20-34
 - reserved area 4-24
 - sbpage structure 4-26
 - starting mirroring 19-22
 - structure 4-24
 - temporary 1-123, 19-12, 20-34
 - creating 19-12
 - user-defined data statistics 1-148
- SBSPACETEMP configuration parameter 1-123, 19-12
- Schedule Manager for Enterprise Replication 20-148
- Scheduler
 - relationship between tables 3-1
 - sensor historical data results tables 3-8
- scheduler argument 21-101
- Scheduler group names
 - ph_group table 3-5
- scheduler lmm disable argument 21-104
- scheduler lmm enable argument 21-101
- Scheduler messages
 - ph_alert table 3-5
- Scheduler task information
 - ph_run table 3-4
 - ph_task table 3-2
- Scheduler task thresholds
 - ph_threshold table 3-8
- Screen reader
 - reading syntax diagrams G-1
- Scripts
 - concdr.sh A-5
 - ex_alarm.sh 1-27, C-1
 - log_full 1-27, C-1
 - no_log 1-27
 - onshutdown 9-2
 - revcdr.sh A-10
- SD secondary server
 - event alarms and messages C-56
- SDS_ENABLE configuration parameter 1-124

- SDS_LOGCHECK configuration parameter 1-124
- SDS_PAGING configuration parameter 1-125
- SDS_TEMPDBS configuration parameter 1-125
- SDS_TIMEOUT configuration parameter 1-127
- security
 - local 1-127
- SECURITY_LOCALCONNECTION configuration parameter 1-127
- SEQ_CACHE_SIZE configuration parameter 1-128
- Server shut down
 - troubleshooting event alarms C-50
- SERVENUM configuration parameter 1-128
- Session information
 - global transactions 20-217
 - SMI tables 2-30, 2-35
- Sessions
 - limiting 1-83
- set chunk argument 21-105
- set dataskip argument 21-106
- SET EXPLAIN statement
 - setting dynamically 14-28
- set index compression argument 21-106
- set onconfig memory argument 21-107
- set onconfig permanent argument 21-108
- set sbspace accesstime argument 21-109
- set sbspace avg_lo_size argument 21-110
- set sbspace logging argument 21-111
- set sql tracing argument 21-111
- set sql tracing database argument 21-113
- set sql tracing session argument 21-114
- set sql tracing user argument 21-115
- set sql user tracing argument 21-115
- SET STATEMENT CACHE statement 1-142, 14-13
- Setting data replication type 14-7
- severity 5 alarms
 - troubleshooting C-50
- Shared library files A-6
- Shared memory
 - adding segment with onmode 14-2
 - base address 1-129, 1-130
 - buffer, frequency of flushing 1-38
 - buffer, maximum number 1-38
 - changing
 - decision-support parameters 14-11
 - residency with onmode 14-18
 - decision-support parameters 21-76
 - dumps 1-67, 1-69
 - examining with SMI 2-2
 - initializing 12-1
 - monitoring 20-1
 - physical-log buffer 1-109
 - resident portion, flag 1-115
 - saving copy of with onstat 20-21
 - segments, dynamically added, size 1-128
 - SERVENUM configuration parameter 1-128
 - size displayed by onstat 20-23
 - virtual segment, initial size 1-132
- shared memory dump file
 - using onstat commands 20-25
- SHMADD configuration parameter 21-71
 - 64-bit addressing 1-128
 - defined 1-128
- SHMBASE configuration parameter 1-129
- shmfile
 - DUMPSHMEM configuration parameter 1-69
 - shmfile.xxx A-10
- SHMNOACCESS configuration parameter 1-130

- SHMTOTAL configuration parameter 1-130
- SHMVIRT_ALLOCSEG configuration parameter 1-131
- SHMVIRT_SIZE configuration parameter 1-132
- Shortcut keys
 - keyboard G-1
- Shutting down the database server 9-1, 9-2, 14-15, 14-16
- SINGLE_CPU_VP configuration parameter 1-134
- Size
 - chunk 19-2, 19-3
 - index fragments 4-8
 - metadata 19-11
 - offset 19-5, 19-7, 19-11, 19-16
- sm_versions.std file A-10
- Smart large objects 1-148
 - buffer pool 1-38
 - cleaning up references 19-18
 - default name 1-122
 - logging 19-14
 - logical-log records 5-14
- SMX
 - compression 1-135
- SMX_COMPRESS configuration parameter 1-135
- Snapshot
 - clone a snapshot of a database server 18-1
- snmpd.log file A-10
- SP_AUTOEXPAND configuration parameter 1-136
- SP_THRESHOLD configuration parameter 1-136
- SP_WAITTIME configuration parameter 1-137
- Specify
 - modified pages, percentage
 - LRU queue 16-4
- Specifying pathname 19-2
- SQL administration API
 - admin() functions 21-1
 - remote administration 21-1
 - task() functions 21-1
- SQL administration API functions
 - add bufferpool argument 21-14
 - add chunk argument 21-15
 - add log argument 21-16
 - add memory argument 21-17
 - add mirror argument 21-18
 - admin() 21-1
 - argument size 21-2
 - return codes 21-2
 - syntax rules 21-1
 - alter chunk argument 21-19
 - alter logmode argument 11-1, 21-20
 - alter plog argument 21-21
 - archive fake argument 21-21
 - arguments by functional category 21-4
 - cdr argument 21-22
 - check data argument 21-23
 - check extents argument 21-24
 - check partition argument 21-24
 - checkpoint argument 21-25
 - clean sbspace argument 21-26
 - compress argument 21-126
 - compression purge_dictionary argument 21-132
 - create blobspace argument 21-26
 - create blobspace from storagepool argument 21-28
 - create chunk argument 21-29
 - create chunk from storagepool argument 21-30
 - create database argument 21-31
 - create dbspace argument 21-32
 - create dbspace from storagepool argument 21-33
 - create sbspace argument 21-34

- SQL administration API functions (*continued*)
 - create sbspace from storagepool argument 21-35
 - create sbspace with accesstime argument 21-36
 - create sbspace with log argument 21-37
 - create tempdbspace argument 21-38
 - create tempdbspace from storagepool argument 21-39
 - create tempsbspace argument 21-39
 - create tempsbspace from storagepool argument 21-41
 - create_dictionary argument 21-126
 - drop blobspace argument 21-43
 - drop blobspace to storagepool argument 21-44
 - drop chunk argument 21-44
 - drop chunk to storagepool argument 21-45
 - drop database argument 21-46
 - drop dbspace argument 21-46
 - drop dbspace to storagepool argument 21-47
 - drop log argument 21-47
 - drop sbspace argument 21-49
 - drop sbspace to storagepool argument 21-49
 - drop tempdbspace argument 21-50
 - drop tempdbspace to storagepool argument 21-50
 - drop tempsbspace to storagepool argument 21-51
 - estimate_compression argument 21-126
 - command output 21-131
 - file status 21-51
 - for compressing data 21-125, 21-126
 - fragment argument 21-126
 - ha make primary argument 21-52
 - ha rss add argument 21-54
 - ha rss argument 21-53
 - ha rss change argument 21-55
 - ha rss delete argument 21-56
 - ha sds clear argument 21-56
 - ha sds primary argument 21-57
 - ha sds set argument 21-58
 - ha set idxauto argument 21-59
 - ha set ipl argument 21-59
 - ha set primary argument 21-60
 - ha set secondary argument 21-61
 - ha set standard argument 21-61
 - ha set timeout argument 21-62
 - message log delete 21-63
 - message log rotate 21-64
 - message log truncate 21-64
 - modify chunk extend argument 21-65
 - modify chunk extendable argument 21-66
 - modify chunk extendable off argument 21-67
 - modify space expand argument 21-68
 - modify space sp_sizes argument 21-69
 - onbar argument 21-92
 - onmode and a arguments 21-71
 - onmode and c arguments 21-72
 - onmode and C arguments 21-73
 - onmode and d arguments 21-74
 - onmode and D arguments 21-76
 - onmode and e arguments 21-76
 - onmode and F arguments 21-77
 - onmode and j arguments 21-78
 - onmode and l arguments 21-79
 - onmode and m arguments 21-79
 - onmode and M arguments 21-80
 - onmode and n arguments 21-81
 - onmode and O arguments 21-81
 - onmode and p arguments 21-82
 - onmode and Q arguments 21-83
 - onmode and r arguments 21-84
 - onmode and S arguments 21-85

SQL administration API functions *(continued)*

- onmode and W arguments 21-85
- onmode and wf arguments 21-87
- onmode and wm arguments 21-88
- onmode and Y arguments 21-90
- onmode and z arguments 21-91
- onmode and Z arguments 21-91
- onmode, wm, and AUTO_LRU_TUNING arguments 21-89
- onsmsync argument 21-93
- ontape archive argument 21-95
- print error argument 21-96
- print file info argument 21-96
- print partition argument 21-97
- rename space argument 21-98
- repack argument 21-126
- repack_offline argument 21-126
- reset sysadmin 21-99
- restart listen argument 21-100
- scheduler 21-101
- scheduler Imm enable 21-101, 21-104
- set chunk argument 21-105
- set dataskip argument 21-106
- set index compression argument 21-106
- set onconfig memory argument 21-107
- set onconfig permanent argument 21-108
- set sbospace accesstime argument 21-109
- set sbospace avg_lo_size argument 21-110
- set sbospace logging argument 21-111
- set sql tracing argument 21-111
- set sql tracing database argument 21-113
- set sql tracing session argument 21-114
- set sql tracing user argument 21-115
- set sql user tracing argument 21-115
- shrink argument 21-126
- start listen argument 21-116
- start mirroring argument 21-117
- stop listen argument 21-117
- stop mirroring argument 21-118
- storagepool add argument 21-119
- storagepool delete argument 21-121
- storagepool modify argument 21-122
- storagepool purge argument 21-124
- table argument 21-126
- task() 21-1
 - argument size 21-2
 - return codes 21-2
 - syntax rules 21-1
- uncompress argument 21-126
- uncompress_offline argument 21-126

SQL profile

- information 2-40

SQL statement cache

- enabling 1-142
- enabling the cache 14-13
- flushing the cache 14-13
- inserting
 - key-only entries 1-144, 14-24
 - qualified statements 1-144, 14-24
- limiting the cache size 1-144
- memory pools 1-145
- specifying number of hits 1-143, 14-24
- turning off the cache 14-13
- turning on the cache 1-142, 14-13

SQL statements

- SET STATEMENT CACHE 1-142, 14-13
- UPDATE STATISTICS 1-149

SQL_LOGICAL_CHAR configuration parameter 1-138

SQLCA, warning flag when fragment skipped during query 1-43

sqlhosts file or registry

- defined A-10
- multiple dbservernames 1-45

sqlmux, multiplexed connections in NETTYPE configuration parameter 1-98

SQLTRACE configuration parameter 1-139

SSL_KEYSTORE_LABEL configuration parameter 1-140

STACKSIZE configuration parameter 1-141

STAGEBLOB configuration parameter 1-142

standards xxv

start listen argument 21-116

start mirroring argument 21-117

Starting database server with oninit 12-1

Starting the database server online 14-15, 14-16

STATCHANGE configuration parameter 1-142

STMT_CACHE configuration parameter 1-142

STMT_CACHE environment variable 14-13

STMT_CACHE environment variables 1-142

STMT_CACHE_HITS configuration parameter 1-143, 14-24

STMT_CACHE_NOLIMIT configuration parameter 1-144

STMT_CACHE_NUMPOOL configuration parameter 1-145

STMT_CACHE_SIZE configuration parameter 1-145

stop listen argument 21-117

stop mirroring argument 21-118

STOP_APPLY configuration parameter 1-145

Storage manager

- xbsa.messages log A-11

Storage pool

- adding an entry 21-119
- create a blobospace from 21-28
- create a chunk from 21-30
- create a dbospace from 21-33
- create a temporary dbospace from 21-39
- create a temporary sbospace from 21-41
- create an sbospace from 21-35
- deleting an entry 21-121
- modifying an entry 21-122
- return space to 21-44, 21-45, 21-47, 21-49, 21-50, 21-51

STORAGE_FULL_ALARM configuration parameter 1-147

storagepool table 3-9

stores_demo

- demonstration database xxiv

superstores_demo

- demonstration database xxiv

Symbolic links

- using with shared libraries A-6

Syntax diagrams

- reading in a screen reader G-1

sysadmin database 3-1, 21-1

Results tables 3-8

storagepool table 3-9

tables

- command_history 3-8
- ph_alert 3-5
- ph_group 3-5
- ph_run 3-4
- ph_task 3-2
- ph_threshold 3-8

sysadinfo table 2-7

SYSALARMPROGRAM configuration parameter 1-147

sysaudit table 2-7

syscheckpoint table 2-8

syschkio table 2-8

syschunks table 2-9

- syscluster table 2-10
- syscmsm table 2-11
- syscmsmsla table 2-12
- syscmsmtab table 2-12
- syscompdicts view 2-13
- syscompdicts_full table 2-13
- sysconfig table 2-14
- sysdatabases table 2-14
- sysdbslocale table 2-15
- sysdbspaces table 2-15
- sysdri table 2-16
- sysdual table 2-17
- sysenv table 2-17
- sysenvses table 2-17
- sysessions table 2-35
- sysextents table 2-17
- sysextspaces table 2-18
- sysfeatures 2-18
- sysfragdist system catalog table 1-148
- sysha_lagtime table 2-19
- sysha_type table 2-20
- sysha_workload table 2-20
- sysipl table 2-21
- syslocks table 2-21
- syslogfil table 2-23
- syslogs table 2-22
- sysmaster database
 - defined 2-1
 - failure to build A-5
 - functionality of 2-1
 - initialization 12-1
 - list of topics covered by 2-2
 - SMI tables 2-2
 - space required to build 2-1
 - sysextspaces 2-18
 - types of tables 2-1
 - warning 2-1
 - when created 2-1
- sysmaster tables
 - syssqexplain 2-37
- sysmgminfo table 2-23
- sysnetclienttype table 2-24
- sysnetglobal table 2-24
- sysnetworkio table 2-25
- sysonlinelog table 2-25
- sysprofile table 2-26
- sysproxyagents table 2-27
- sysproxydistributors table 2-28
- sysproxysessions table 2-28
- sysproxytnops table 2-29
- sysproxytxns table 2-29
- sysptprof table 2-30
- sysrepevtreg table 2-31
- sysrepstats table 2-31
- sysrsslog table 2-34
- SYSSBPACENAME configuration parameter 1-148
- sysscblst table 2-34
- sysseappinfo table 2-34
- sysseprof table 2-35
- sysstm table 2-37
- sysstmxes table 2-37
- syssqexplain
 - sysmaster table 2-37
- syssqltrace table 2-38
- syssqltrace_info table 2-40
- syssqltrace_iter table 2-40
- syssrcss table 2-40
- syssrcsds table 2-41
- systabnames table 2-41
- System catalog tables
 - disk space allocation 4-27
 - listing 8-3
 - oncheck -cc 8-8
 - oncheck -pc 8-8
 - sysdistrib 1-148
 - sysfragdist 1-148
 - sysfragments table 4-6
 - systracees B-1
 - systracemsgs B-1
 - tracking 4-27
 - tracking a new database 4-27
 - tracking a new table 4-28
- system page size, specifying 1-40
- System-monitoring interface
 - accessing SMI tables 2-3
 - defined 2-1
 - list of SMI tables 2-4
 - locking 2-4
 - obtaining onstat information 2-45
 - SPL 2-4
 - systabpaghdrs table 2-3
 - tables
 - defined 2-2
 - list of supported 2-4
 - sysadinfo 2-7
 - sysaudit 2-7
 - syscheckpoint 2-8
 - syschkio 2-8
 - syschunks 2-9
 - syscluster 2-10
 - syscompdicts 2-13
 - syscompdicts_full 2-13
 - sysconfig 2-14
 - sysdatabases 2-14
 - sysdbslocale 2-15
 - sysdbspaces 2-15
 - sysdri 2-16
 - sysdual 2-17
 - sysenv 2-17
 - sysenvses 2-17
 - sysextents 2-17
 - sysextspaces 2-18
 - sysfeatures 2-18
 - sysha_lagtime 2-19
 - sysha_type 2-20
 - sysha_workload 2-20
 - sysipl 2-21
 - syslocks 2-21
 - syslogfil 2-23
 - syslogs 2-22
 - sysmgminfo 2-23
 - sysnetclienttype 2-24
 - sysnetglobal 2-24
 - sysnetworkio 2-25
 - sysonlinelog 2-25
 - sysprofile 2-26
 - sysproxyagents 2-27
 - sysproxydistributors 2-28
 - sysproxysessions 2-28
 - sysproxytnops 2-29
 - sysproxytxns 2-29
 - sysptprof 2-30
 - sysrepevtreg 2-31
 - sysrepstats 2-31

System-monitoring interface *(continued)*

- tables *(continued)*
 - sysrsslog 2-34
 - sysscblst 2-34
 - sysseappinfo 2-34
 - sysseprof 2-35
 - sysseptions 2-35
 - sysmx 2-37
 - sysmxses 2-37
 - sysqltrace 2-38
 - sysqltrace_info 2-40
 - sysqltrace_iter 2-40
 - syssrcss 2-40
 - sysrcsds 2-41
 - systabnames 2-41
 - systhreads 2-42
 - sysrgrss 2-42
 - sysrgrsds 2-43
 - sysvpprof 2-43
- triggers 2-3
- using SELECT statements 2-3
- viewing tables with dbaccess 2-3
- views 2-3, 2-4
- systhreads table 2-42
- sysracees table B-1
- sysracemsgs table B-1
- sysrgrss table 2-42
- sysrgrsds table 2-43
- sysutil tables 2-6
- sysvpprof table 2-43

T

Table

- creating, what happens on disk 4-27, 4-28
- displaying allocation information 8-19
- extent size doubling 4-11
- lock mode 1-52
- monitoring with SMI 2-41
- pseudotables 2-2
- SMI tables 2-2
- temporary
 - effects of creating 4-28
 - message reporting cleanup E-16
- tables
 - defragmenting 21-41
 - sysrepevtreg 2-31
 - sysrepstats 2-31
- tail -f command E-1
- task() functions 21-1
 - add bufferpool argument 21-14
 - add chunk argument 21-15
 - add log argument 21-16
 - add memory argument 21-17
 - add mirror argument 21-18
 - alter chunk argument 21-19
 - alter logmode argument 11-1, 21-20
 - alter plog argument 21-21
 - archive fake argument 21-21
 - argument size 21-2
 - arguments by functional category 21-4
 - cdr argument 21-22
 - check data argument 21-23
 - check extents argument 21-24
 - check partition argument 21-24
 - checkpoint argument 21-25
 - clean sbpace argument 21-26

task() functions *(continued)*

- compress argument 21-126
- compression purge_dictionary argument 21-132
- create blobspace argument 21-26
- create blobspace from storagepool argument 21-28
- create chunk argument 21-29
- create chunk from storagepool argument 21-30
- create database argument 21-31
- create dbspace argument 21-32
- create dbspace from storagepool argument 21-33
- create sbpace argument 21-34
- create sbpace from storagepool argument 21-35
- create sbpace with accesstime argument 21-36
- create sbpace with log argument 21-37
- create tempdbspace argument 21-38
- create tempdbspace from storagepool argument 21-39
- create tempsbpace argument 21-39
- create tempsbpace from storagepool argument 21-41
- create_dictionary argument 21-126
- drop blobspace argument 21-43
- drop blobspace to storagepool argument 21-44
- drop chunk argument 21-44
- drop chunk to storagepool argument 21-45
- drop database argument 21-46
- drop dbspace argument 21-46
- drop dbspace to storagepool argument 21-47
- drop log argument 21-47
- drop sbpace argument 21-49
- drop sbpace to storagepool argument 21-49
- drop tempdbspace argument 21-50
- drop tempdbspace to storagepool argument 21-50
- drop tempsbpace to storagepool argument 21-51
- estimate_compression argument 21-126
 - command output 21-131
- file status 21-51
- for compressing data 21-125, 21-126
- fragment argument 21-126
- ha make primary argument 21-52
- ha rss add argument 21-54
- ha rss argument 21-53
- ha rss change argument 21-55
- ha rss delete argument 21-56
- ha sds clear argument 21-56
- ha sds primary argument 21-57
- ha sds set argument 21-58
- ha set idxauto argument 21-59
- ha set ipl argument 21-59
- ha set primary argument 21-60
- ha set secondary argument 21-61
- ha set standard argument 21-61
- ha set timeout argument 21-62
- message log delete 21-63
- message log rotate 21-64
- message log truncate 21-64
- modify chunk extend argument 21-65
- modify chunk extendable argument 21-66
- modify chunk extendable off argument 21-67
- modify space expand argument 21-68
- modify space sp_sizes argument 21-69
- onbar argument 21-92
- onmode and a arguments 21-71
- onmode and c arguments 21-72
- onmode and C arguments 21-73
- onmode and d arguments 21-74
- onmode and D arguments 21-76
- onmode and e arguments 21-76
- onmode and F arguments 21-77

- task() functions *(continued)*
 - onmode and j arguments 21-78
 - onmode and l arguments 21-79
 - onmode and m arguments 21-79
 - onmode and M arguments 21-80
 - onmode and n arguments 21-81
 - onmode and O arguments 21-81
 - onmode and p arguments 21-82
 - onmode and Q arguments 21-83
 - onmode and r arguments 21-84
 - onmode and S arguments 21-85
 - onmode and W arguments 21-85
 - onmode and wf arguments 21-87
 - onmode and wm arguments 21-88
 - onmode and Y arguments 21-90
 - onmode and z arguments 21-91
 - onmode and Z arguments 21-91
 - onmode, wm, and AUTO_LRU_TUNING arguments 21-89
 - onsmsync argument 21-93
 - ontape archive argument 21-95
 - print error argument 21-96
 - print file info argument 21-96
 - print partition argument 21-97
 - rename space argument 21-98
 - repack argument 21-126
 - repack_offline argument 21-126
 - reset sysadmin argument 21-99
 - restart listen argument 21-100
 - return codes 21-2
 - scheduler argument 21-101
 - scheduler lmm disable argument 21-104
 - scheduler lmm enable argument 21-101
 - set chunk argument 21-105
 - set dataskip argument 21-106
 - set index compression argument 21-106
 - set onconfig memory argument 21-107
 - set onconfig permanent argument 21-108
 - set sbospace accesstime argument 21-109
 - set sbospace avg_lo_size argument 21-110
 - set sbospace logging argument 21-111
 - set sql tracing argument 21-111
 - set sql tracing database argument 21-113
 - set sql tracing session argument 21-114
 - set sql tracing user argument 21-115
 - set sql user tracing argument 21-115
 - shrink argument 21-126
 - start listen argument 21-116
 - start mirroring argument 21-117
 - stop listen argument 21-117
 - stop mirroring argument 21-118
 - storagepool add argument 21-119
 - storagepool delete argument 21-121
 - storagepool modify argument 21-122
 - storagepool purge argument 21-124
 - syntax rules 21-1
 - table argument 21-126
 - uncompress argument 21-126
 - uncompress_offline argument 21-126
- Tblspace
 - displaying (onstat -t or -T) 20-22, 20-211
 - monitoring
 - blspace statistics 1-149
 - with SMI 2-30
 - number 4-5, 20-212
 - table fragment 4-6, 4-15
- Tblspace number
 - defined 4-5
 - elements 4-6
 - includes dbspace number 4-5
 - table fragment 4-6
- Tblspace tblspace
 - bitmap page 4-6
 - location in a chunk 4-2
 - location in root dbspace 4-1
 - tracking new tables 4-28
- TBLSPACE_STATS configuration parameter 1-149
 - with onstat -g ppf 20-132
- TBLTBLFIRST configuration parameter 1-150
- TBLTBLNEXT configuration parameter 1-150
- Template
 - ac_config.std file A-4
 - onconfig.std file A-8
- Temporary dbspace
 - creating with onspaces 19-7, 19-10
 - DBSPACETEMP configuration parameter 19-10
- Temporary sbospace
 - creating with onspaces 19-12
 - onstat -d 20-34
 - SBSPACETEMP configuration parameter 1-123
- temporary sbospaces
 - creating with onspaces 19-11
- Temporary smart large object
 - default sbospace 1-123
 - SBSPACETEMP configuration parameter 1-123
- Temporary tables
 - DBSPACETEMP configuration parameter 1-48
 - extent size doubling 4-11
 - rules for use 1-48
- TEMPTAB_NOLOG configuration parameter 1-151
- TEXT and BYTE data
 - blob descriptor 4-13, 4-22
 - modifying storage 4-22
 - page descriptor 4-22
 - size limitations 4-23
 - storage on disk 4-21, 4-22
 - updating 4-22
 - when modified 4-22
 - when written 4-22
- thread
 - status 20-185
 - wait statistics 20-185
- Threads
 - Buffers
 - page-type codes 20-217
 - onstat -X usage 20-22, 20-217
- Tightly-coupled mode 20-217
- Time stamp
 - blobspace blobpage 4-23
 - defined 4-27
- Time-out condition 20-40
- Trace B-1
- Trace message B-1
- Tracepoints B-1
- Transaction manager
 - loosely-coupled mode 20-217
 - tightly-coupled mode 20-217
- Transaction Replicate Group E-62
- Transaction survival 1-78
- Transaction-logging database property 2-14
- Transactions
 - heterogeneous commit 1-81
 - kill with onmode -Z 14-29

- Transactions *(continued)*
 - pending 20-215
 - XID 20-217
- Trapping errors with onmode B-1
- Troubleshooting
 - severity 5 event alarms C-50
- TRUNCATE 5-3
- TRUNCATE log record 5-3
- Truncate table messages E-68
- Tuning
 - large number of users 1-98
 - use of NETTYPE configuration parameter 1-98
- Tuning recommendations 20-61
- Turning on SQL statement cache 14-13
- Two-phase commit protocol, killing distributed transactions 14-29
- TXTIMEOUT configuration parameter
 - defined 1-151
 - onmode utility 14-29

U

- UDINSERT
 - record subtype 5-14
- UDUPAFT
 - record subtype 5-14
- UDUPBEF
 - record subtype 5-14
- UDWRITE
 - record subtype 5-14
- Unblocking database server 14-4
- Unbuffered disk space
 - UNIX 19-2
 - Windows 19-2, 19-5, 19-7
- Unbuffered-logging database property 2-14
- UNIX
 - buffered disk space 19-2
 - interrupt signal 6-1
 - unbuffered disk space 19-2
 - using onspaces 19-1
- UNSECURE_ONSTAT configuration parameter 1-152
- UPDATABLE_SECONDARY configuration parameter 1-152
- UPDATE STATISTICS statement 1-61, 1-64, 1-149
- Updating blob space statistics 20-34
- USELASTCOMMITTED configuration parameter 1-153
- USEOSTIME parameter 1-155
- User connections, monitoring E-25
- User session
 - monitoring with SMI 2-35
 - status codes 20-212
- user-defined data statistics 1-148
- User-defined routines, debugging B-1
- User-defined type
 - data distributions 1-148
- USERMAPPING configuration parameter 1-163
- USTLOW_SAMPLE configuration parameter 1-155
- Utilities 6-1
 - V option 6-1
 - version option 6-1
 - gcore 1-67, 1-68
 - genoncfg 7-1
 - IBM Informix Server Administrator 6-2
 - ifxclone 18-1
 - ON-Monitor utility 1-40, 15-1, 15-4
 - oncheck 8-1, 8-22
 - onclean 9-1
 - oncmsm 10-1

- Utilities *(continued)*
 - ondblog utility 11-1
 - oninit 12-1
 - onlog 13-1
 - onmode 14-1, 14-28
 - onmode and PDQ 20-116
 - onparams 16-1, 16-6
 - onpassword 17-1, 17-3
 - onspaces 19-1, 19-21
 - onstat utility 20-1, 20-219
 - g env option 20-86
 - g mgm option 20-116
 - g seg option 1-128
 - option 6-1

V

- VARCHAR data type
 - 4-bit bit map requirement 4-8, 4-9
 - implications for data row storage 4-14
 - indexing considerations 4-20
 - storage considerations 4-12
- Violations table
 - messages E-4, E-44
- Virtual processors
 - adding or removing with onmode 14-19
 - designating a class 1-157
 - limits 14-19
 - number in
 - AIO class D-8
 - CPU class D-9
 - priority aging D-7
 - processor affinity 1-97
- Visual disabilities
 - reading syntax diagrams G-1
- VP configuration parameter
 - AFF_NPROCS D-2
- VP_MEMORY_CACHE_KB configuration parameter
 - defined 1-156
- VP.servername.nnx file A-11
- VPCLASS configuration parameter
 - AFF_SPROC D-3
 - default values 1-157
 - defined 1-157
 - in ONCONFIG file 1-157
 - NOAGE D-8
 - NUMAIOVPS D-8
 - onmode utility 14-19
 - reserved names 1-158
 - setting maximum virtual processors 1-161
 - setting number of virtual processors 1-160
 - setting processor affinity 1-161
 - user-defined classes 1-159

W

- Warnings
 - buildsmi script 2-2
 - when fragment skipped during query processing 1-43
- Windows
 - adding or removing virtual processors 14-21
 - buffered disk space 19-2
 - unbuffered disk space 19-2, 19-5, 19-7
 - using onspaces 19-1
- WSTATS configuration parameter 1-163

X

xbsa.messages log A-11



Printed in USA

SC27-3530-03



Spine information:

Informix Product Family Informix

Version 11.70

IBM Informix Administrator's Reference

