Exploring the Informix OnLine Utilities

by Lester Knutsen

INFORMIX-OnLine DSA comes with a set of powerful command line utilities that enable you to monitor, tune, and configure your database server. This chapter will focus on eight of these utilities, and present ways to use them to optimize your performance as a database administrator.

The command line utilities and the order in which we will discuss them is as follows:

ONSTAT - shows shared memory and server statistics
ONCHECK - checks and repairs disk space
ONMODE - changes OnLine's operating mode
ONLOG - logical log debugging tool
ONINIT - initialize and start up the database server
ONSPACES - configure dbspaces and chunks
ONPARAMS - configure logs
ONTAPE - backup and restore utility
ONLOAD - loads databases and tables
ONUNLOAD - unloads databases and tables

ONSTAT - shows OnLine server statistics

ONSTAT is the command line utility that gets the most usage. It reads OnLine's shared memory structures and provides lots of useful information about the state of your server. It does not place any locks on shared memory structure and uses very little overhead, so you can use it at any time. The information is current at the time the command is issued, and the data can change as you are using the command.

There are more options for ONSTAT than any other Informix utility. Many of the options are debugging parameters that are not well documented and don't make sense to the average DBA. There are also many very useful options that help you manage your OnLine server. We will focus on the more useful options in this chapter. For a complete list of the syntax and all the options, see Figure 1. In INFORMIX-OnLine DSA the ONSTAT command has been greatly enhanced with a new set of monitoring and debugging options. These options all start with '-g' and are listed in Figure 2.

Figure 1: ONSTAT syntax and options

```
onstat [-abcdfghklmpstuxzBCDFRX][-I] [-r seconds] [-o file] [infile]
          -a        Print all information
          -b        Print buffers
          -c        Print configuration file
          -d        Print DBspaces and chunks
          -f        Print dataskip status
          -g        New Monitoring subcommands (default: all). See Figure 2 for all options
          -i        Interactive mode
          -k        Print locks
          -l        Print logging
          -m        Print message log
          -p        Print profile
          -s        Print latches
          -t        Print TBLspaces
          -u        Print user threads
          -x        Print transactions
          -z        Zero profile counts
          -B        Print all buffers
          -C        Print btree cleaner requests
          -D        Print DBspaces and detailed chunk stats
          -F        Print page flushers
          -R        Print LRU queues
          -X        Print entire list of sharers and waiters for buffers
```

-r	Repeat options every n seconds (default: 5)
-o	Put shared memory into specified file (default: onstat.out)
infile	Use infile to obtain shared memory information
-	Displays OnLine mode

Figure 2: ONSTAT new options (-g) syntax

onstat -g [options from list below]

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 class
ppf	[<partition number> | 0] Print partition profiles
tpf	[<tid> | 0] Print thread profiles
ntu	Print net user thread profile information
ntt	Print net user thread access times
ntm	Print net message information
ntd	Print net dispatch information
nss	<session id> Print net shared memory status
nsc	<client id> Print net shared memory status
nsd	Print net shared memory data
sts	Print max and current stack sizes
dic	Print dictionary cache information
qst	Print queue statistics
wst	Print thread wait statistics
ses	<session id> Print session information
sql	<session id> Print sql information
dri	Print data replication information
pos	Print /INFORMIXDIR/etc/.infos.DBSERVERNAME file
mgm	Print mgm resource manager information
ddr	Print DDR log post processing information

There are too many commands to cover all of them in this chapter. Instead we will focus on the key commands and the ones that are most useful to a DBA in monitoring your server.

Current status of OnLine: onstat -

The command "onstat -" prints out a one line message indicating the current status of your server. This is a quick way to get a status update. Figure 3 has an example output.

Figure 3: Current status: onstat -

```
lester@merlin >onstat -

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 11:54:44 -- 10656 Kbytes
```

This tells the INFORMIX-OnLine version, what mode the server is in, how long it has been up and running, and how much memory it is using. If the server was down, you will get an error message saying "`shared memory not initialized`" like the one in Figure 4.

Figure 4: Current status when OnLine is down

```
lester@merlin >onstat -
shared memory not initialized for INFORMIXSERVER 'merlindb713'
lester@merlin >
```

Database server profile: onstat -p

The “-p” displays the basic I/O and performance profile of your system. Figure 5 contains example output. These statistics are since the server was last rebooted, or when the statistics were last reset with the “onstat -z” option.

Figure 5: Server profile: onstat -p

```
lester@merlin >onstat -p

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:03:37 -- 10656
Kbytes

Profile
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
215537   2098656  5208789  95.86   178116   179527   2883605  93.82

isamtot  open     start    read     write    rewrite  delete   commit   rollbk
6955097  2172     2858     2312158  2305564  129      43       22238    0

ovlock   ovuserthread ovbuff   usercpu  syscpu   numckpts flushes
0        0            168      6625.92  722.70   35       4320

bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
440      0        2331106  0        0        29       5        66

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
836      0        16       833        36
```

Some of the key elements of this option are:

Reads %cached - This is the percent of your reads that are using OnLine’s buffers instead of accessing disk drives because the records are already in memory. The goal is to have 95% of your reads come from the buffers. If this number is below 95% you may need to increase the BUFFERS parameter in your ONCONFIG file.

Writes %cached - This is the percent of writes that are using your buffers. The goal here is to have 85% or more of your write activity use the buffers. The one exception is during large data loads. The BUFFERS parameter in your ONCONFIG file will effect this value. Be careful when you increase the BUFFERS parameter - if you make the BUFFERS too large this will take memory away from other processes and may slow down your whole system. As you increase BUFFERS, you need to monitor swapping and paging of your operating system.

ovlock - This should be zero. Any other number indicates you have run out of locks since the system was last reset. Increase the LOCKS parameter in the ONCONFIG file.

ovuserthread - This should be zero. This value is increased each time a user tries to connect and the number of current users exceeds the maximum number of user threads set in the ONCONFIG file. The maximum number of user threads is the third value of the NETTYPE parameter in the ONCONFIG file.

ovbuff - This should be zero. This value is increased every time OnLine tries to acquire more buffers than are set by the BUFFER parameter in the ONCONFIG file.

bufwaits - This should be zero. This indicates the number of times a user thread has waited for a BUFFER.

lokwaits - This should be zero. This indicates the number of times a user thread has waited for a LOCK.

deadlks - This should be zero. This indicates the number of times a deadlock was detected and prevented.

dltouts - This should be zero. This indicates the number of times a distributed deadlock was detected.

Display message log file: onstat -m

This option displays the last lines of the OnLine message log. This is the message file that contains all messages about your server and is a key component of your system to monitor. Figure 6 contains an example. This is a quick way to see the last 20 messages in your log file.

Figure 6: Display message log file: onstat -m

```
lester@merlin >onstat -m

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:41:12 -- 10656
Kbytes

Message Log File: /u3/informix7/online1.log
21:05:15  Checkpoint Completed:  duration was 8 seconds.
21:05:43  Checkpoint Completed:  duration was 6 seconds.
21:10:58  Checkpoint Completed:  duration was 7 seconds.
21:16:06  Checkpoint Completed:  duration was 7 seconds.
21:21:13  Checkpoint Completed:  duration was 7 seconds.
21:26:20  Checkpoint Completed:  duration was 7 seconds.
21:31:28  Checkpoint Completed:  duration was 7 seconds.
21:36:36  Checkpoint Completed:  duration was 8 seconds.
21:41:43  Checkpoint Completed:  duration was 7 seconds.
21:46:51  Checkpoint Completed:  duration was 8 seconds.
21:52:00  Checkpoint Completed:  duration was 9 seconds.
21:57:09  Checkpoint Completed:  duration was 8 seconds.
22:00:42  Logical Log 20 Complete.
22:00:43  Process exited with return code 1: /bin/sh /bin/sh -c
/u3/informix7/lo
g_full.sh 2 23 "Logical Log 20 Complete." "Logical Log 20 Complete."
22:02:17  Checkpoint Completed:  duration was 8 seconds.
```

Note: I like to have the OnLine log file always display in one of my windows on screen. The trick to doing this is to use the UNIX "tail" command with the "-f" option. This continually reads the last lines of a file as it is appended to. On my system I run the following command to continually monitor this log:

```
        tail -f $INFORMIXDIR/online.log
```

User status: onstat -u

The ONSTAT option to monitor what your users are doing is"-u". Figure 7 shows example output from this command. The key field is "sessid". This identifies the users session ID that OnLine uses to track the user internally. This is the number you need to know if you need to kill a user's session. (See "onmode -z" later in this chapter)

Figure 7: User status: onstat -u

```
lester@merlin >onstat -u

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:52:40 -- 10656
Kbytes

Userthreads
address  flags   sessid   user     tty      wait     tout locks nreads
nwrites
a2d0018  ---P--D 1        informix -        0        0    0     84       250
a2d0458  ---P--F 0        informix -        0        0    0     0        177701
a2d0898  ---P--B 8        informix -        0        0    0     0        0
a2d1558  ---P--D 12       informix -        0        0    0     0        0
a2d2218  Y--P--- 264      lester   4        a35f190  0    1     35063    14836
 5 active, 128 total, 17 maximum concurrent
```

The "flags" column gives you some idea of what a user is doing. The following are the critical flags, based on position within the flag field:

Flags in position 1
- B - Waiting on a buffer
- C - Waiting on a checkpoint
- G - Waiting on a logical log buffer write
- L - Waiting on a lock
- S - Waiting on a mutex
- T - Waiting on a transaction
- Y - Waiting on a condition
- X - Waiting on a transaction rollback

Flags in position 2
- * - Transaction active during I/O error

Flags in position 3
- A - Dbspace backup thread
- B - Begin work
- P - Prepared for commit work
- X - TP/XA prepeared for commit work
- C - Committing work
- R - Rolling back work
- H - Heuristically rolling back work

Flags in position 4
- P - Primary thread for a session

Flags in position 5
- R - Reading call
- X - Transaction is committing

Flags in position 6
- None

Flags in position 7
- B - Btree cleaner thread
- C - Cleanup of terminated user
- D - Daemon thread
- F - Page flusher thread
- M - ON-Monitor user thread

Logical Logs status: onstat -l

The “-l” option to ONSTAT displays the current status of your logical logs. Figure 8 shows an example display. One problem with this display is that it does not really show you which logs are ready to be reused. In the 5.X versions of OnLine, as soon as a log was backed up and had no open transactions it would be marked as free with an “F” in the flags column. In the current versions of OnLine, logs are not marked as free until right before they need to be reused. (See chapter 26 for an SMI script to show logs that really are free.) One way of using ONSTAT to tell which logs can be reused is to use “onstat -l” with “onstat -x” to display all active sessions. See the next section on “onstat -x”.

Figure 8: Logical Logs status: onstat -l

```
Physical Logging
Buffer bufused  bufsize  numpages numwrits pages/io
  P-1  0        16       236      60        3.93
      phybegin physize  phypos   phyused  %used
      10003f   1000     967      0        0.00

Logical Logging
Buffer bufused  bufsize  numrecs  numpages numwrits recs/pages pages/io
  L-3  0        16       90303    1522     275      59.3       5.5

address  number   flags    uniqid   begin        size     used    %used
a1ee3e4  1        U-B----  13       100427        500      500   100.00
a1ee400  2        U-B----  14       10061b        500      500   100.00
a1ee41c  3        U-B----  15       10080f        500      500   100.00
a1ee438  4        U-B----  16       100a03        500      500   100.00
a1ee454  5        U-B----  17       100bf7        500      432    86.40
a1ee470  6        U-B----  18       100deb        500      500   100.00
a1ee48c  7        U-B----  19       100fdf        500      500   100.00
a1ee4a8  8        U-B----  20       1011d3        500      500   100.00
a1ee4c4  9        U---C-L  21       1013c7        500       23     4.60
a1ee4e0  10       U-B----  10       1015bb        500      500   100.00
a1ee4fc  11       U-B----  11       1017af        500      500   100.00
a1ee518  12       U-B----  12       1019a3        500      500   100.00
```

The flags column provides status information about each log. The flags are:

A - Newly added, must run an archive before they can be used
B - Backed up to tape or “/dev/null”
C - Current logical log file
F - Free and available for use. You will rarely see this flag as logs are not marked as free until right before they are needed.
L - Last checkpoint is in this logical log
U - Used logical log, it may be free if it is backed up and contains no active transactions.

Display transactions: onstat -x

This option displays all current transactions. The most useful column is "log begin". This tells you in which logical log a transaction started. This may be used with the "onstat -l" command to determine which logs are free and may be reused. Find the earliest logical log number in the column "log begin". This tells you which logical log has the earliest active transaction. Any logical logs that are backed up before the log with the earliest transaction will be automatically reused by OnLine.

Figure 9: Transactions status: onstat -x

```
lester@merlin >onstat -x

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 01:21:38 -- 10656 Kbytes

Transactions
address  flags userthread locks log begin isolation retrys coordinator
a2f4018  A---- a2d0018    0     0         COMMIT    0
a2f413c  A---- a2d0458    0     0         COMMIT    0
a2f4260  A---- a2d0898    0     0         COMMIT    0
a2f4384  A---- a2d1118    0     0         NOTRANS   0
a2f44a8  A---- a2d1558    0     0         COMMIT    0
a2f45cc  A-B-- a2d1118    2     21        NOTRANS   0
 6 active, 128 total, 7 maximum concurrent
```

Display locks: onstat -k

The "onstat -k" option will display all active locks. Watch out, this display could be long. If you have a large number of LOCKS defined in your ONCONFIG file and many users you could see thousands of rows from this command. Figure 10 is an example of the display.

Figure 10: Display all locks: onstat -k

```
lester@merlin >onstat -k

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:53:31 -- 10656
Kbytes

Locks
address  wtlist   owner    lklist   type     tblsnum  rowid    key#/bsiz
a103e74  0        a2d2218  0        HDR+S    100002   20a         0
 1 active, 20000 total, 16384 hash buckets
```

Who owns a lock?

The "owner" column lists the address in shared memory of the user who owns a lock. Use this with "onstat -u" to see all users, and compare this with the "address" column to identify username of the owner.

What table is locked?

The “tblsnum” column identifies the table that is being locked. Compare this with the output of the following SQL statement to convert a table’s partnum to hex. This will identify which table is locked.

```
select tabname, hex(partnum) tblsnum
from systables where tabid > 99;
```

This SQL statement will provide you with a list of tables and their associated tblsnum to identify which table has a lock placed on it. Figure 11 contains an example of how to identify which table is locked.

Figure 11: What table is locked

```
1.  Find a list of tblsnum

dbaccess database - <<EOF
      select tabname, hex(partnum) tblsnum
      from systables where tabid > 99;
EOF

database selected

tabname              tblsnum
genjournal          0x0010009E
gjsum               0x0010009F

2.  Find what is locked
onstat -k

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 01:47:38 -- 10656 Kbytes

Locks
address  wtlist   owner    lklist   type     tblsnum  rowid    key#/bsiz
a103e44  0        a2d1118  a103de4  HDR+X    10009f   0            0
 3 active, 20000 total, 16384 hash buckets

3.  Compare tblsnum from step 1 and step 2.  This identifies the table gjsum
as the one that is locked.
```

The tblsnum 100002 has a special meaning. This indicates a database lock. Every user who opens a database will place a shared lock on the database.

Types of locks

The following list the types of locks and how to identify them.

Database	- Lock is placed on tablespace 1000002
Table	- Lock is placed on actual tablespace with rowid of 0
Page	- Lock is placed on tablespace with rowid ending in 00
Row	- Lock is placed on tablespace with actual rowid (not 00)
Byte	- Lock is placed on tablespace/page with size of bytes
Key	- Lock is placed on tablespace hex rowid (starting with f)

Lock type flags

The following lists the lock flags in the “flags” column of “onstat -k”:

HDR	- Header
B	- Bytes lock
S	- Shared lock
X	- Exclusive
I	- Intent
U	- Update
IX	- Intent-exclusive
IS	- Intent-shared
SIX	- Shared, Intent-exclusive

Dbspaces and chunks status: onstat -d

This ONSTAT command shows two very important items - the layout of your dbspaces and disk chunks, and the status of each chunk and dbspace. Print the output of this command and save it. You will need it if you ever have to perform a restore. This identifies each dbspace and chunk that you need to rebuild your system. Figure 12 contains an example output from one of my training systems.

Figure 12: Dbspaces and chunk status: onstat -d

```
lester@merlin >onstat -d

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:54:44 -- 10656
Kbytes

Dbspaces
address  number   flags    fchunk   nchunks  flags    owner    name
a2ce100  1        1        1        1        N        informix rootdbs
a2ce508  2        1        2        1        N        informix dbspace1
a2ce578  3        1        3        1        N        informix dbspace2
a2ce5e8  4        1        4        1        N        informix dbspace3
 4 active, 2047 maximum

Chunks
address  chk/dbs offset   size     free     bpages   flags pathname
a2ce170  1   1   0        250000   62047             PO-   /u3/dev/rootdbs1
a2ce280  2   2   0        10000    9587              PO-   /u3/dev/dbspace1
a2ce358  3   3   0        10000    9947              PO-   /u3/dev/dbspace2
a2ce430  4   4   0        10000    9947              PO-   /u3/dev/dbspace3
 4 active, 2047 maximum
```

This display also shows how much free space each chunk has, and the status of each chunk.

The “flags” for Dbspaces are:

- Position 1
    - M - Mirrored Dbspace
    - N - Not Mirrored Dbspace
- Position 2
    - X - Newly mirrored
    - P - Physical recovery underway
    - L - Logical recovery underway
    - R - Recovery underway
- Position 3
    - B - Blobspace

The “flags” for Chunks are:

- Position 1
    - P - Primary
    - M - Mirror
- Position 2
    - O - On-line
    - D - Down
    - X - Newly mirrored
    - I - Inconsistent
- Position 3
    - B - Blobspace
    - - - Dbspace
    - T - Temporary Dbspace

Dbspaces and chunks I/O: onstat -D

The "onstat -D" option shows I/O by chunk. This is very helpful in performance tuning. Your goal is to spread your reads and writes evenly across all chunks. Figure 13 shows an example where one chunk is utilized for all I/O, and all other chunks are inactive. The I/O is not spread out among chunks, which is not an effective use of disk.

Figure 13: Dbspaces and chunks IO: onstat -D

```
lester@merlin >onstat -D

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:55:09 -- 10656
Kbytes

Dbspaces
address  number   flags    fchunk   nchunks  flags    owner    name
a2ce100  1        1        1        1        N        informix rootdbs
a2ce508  2        1        2        1        N        informix dbspace1
a2ce578  3        1        3        1        N        informix dbspace2
a2ce5e8  4        1        4        1        N        informix dbspace3
 4 active, 2047 maximum

Chunks
address  chk/dbs offset   page Rd  page Wr  pathname
a2ce170  1   1   0        36563    179558   /u3/dev/rootdbs1
a2ce280  2   2   0        3        0        /u3/dev/dbspace1
a2ce358  3   3   0        2        0        /u3/dev/dbspace2
a2ce430  4   4   0        2        0        /u3/dev/dbspace3
 4 active, 2047 maximum
```

Page write status: onstat -F

There are three ways OnLine writes pages from shared memory buffers to disk. Foreground writes occur when OnLine needs a buffer and must interrupt processing to flush buffers to disk to free a buffer. These are the least desirable type of writes. Background writes (LRU Writes) occur when a set percent of the buffers are dirty. This is controlled by the LRU parameters in the ONCONFIG file. These do not interrupt user processing and are the best for interactive systems. Chunk writes occur at checkpoints, and all dirty buffer pages are written to disk. The more dirty pages, the longer a checkpoint will take. Checkpoint writes are sorted and optimized, but the longer a checkpoint is, the longer it will block user activity. Checkpoint writes are best for batch systems. The ONSTAT option to monitor this is "-F". The goal should be to see zero
foreground writes (Fg Writes). Figure 14 contains an example.

Figure 14: Page writes status: onstat -D

```
lester@merlin >onstat -F

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:55:32 -- 10656
Kbytes

Fg Writes     LRU Writes    Chunk Writes
168           172280        5277

address  flusher  state    data
a2d0458  0        I        0        = 0X0
      states: Exit Idle Chunk Lru
```

New monitoring and debugging commands (version 7.X): onstat -g

The “-g” commands are a whole new subset of commands in OnLine version 7. Figure 2 (earlier in this chapter) contains a list of all the -g commands. This section will discuss the most interesting of these.

List all threads: onstat -g ath

This option lists all active threads. Figure 15 shows an example.

Figure 15: List all active threads: onstat -g ath

```
INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:56:09 -- 10656
Kbytes

Threads:
tid     tcb     rstcb    prty    status                   vp-class   name

2       a336b70  0        2      sleeping(Forever)          3lio     lio vp 0
3       a336dd0  0        2      sleeping(Forever)          4pio     pio vp 0
4       a337088  0        2      sleeping(Forever)          5aio     aio vp 0
5       a337340  0        2      sleeping(Forever)          6msc     msc vp 0
6       a337af8  0        2      sleeping(Forever)          7aio     aio vp 1
7       a337e00  a2d0018  4      sleeping(secs: 1)          1cpu     main_loop()
8       a34ab48  0        2      running                    1cpu     sm_poll
9       a34b770  0        2      running                    8tli     tlitcppoll
10      a34bce0  0        2      sleeping(Forever)          1cpu     sm_listen
11      a3c4a28  0        2      sleeping(secs: 2)          1cpu     sm_discon
12      a3c4e58  0        3      sleeping(Forever)          1cpu     tlitcplst
13      a3d0680  a2d0458  2      sleeping(Forever)          1cpu     flush_sub(0)
14      a3d0e40  a2d0898  2      sleeping(secs: 8)          1cpu     btclean
30      a35ea58  a2d1558  4      sleeping(secs: 1)          1cpu     onmode_mon
283     a39ef38  a2d2218  2      cond wait(sm_read)         1cpu     sqlexec
```

List Virtual Processor status: onstat -g sch

This option provides the means to identify which “oninit” UNIX process corresponds to which OnLine server Virtual Processor. When you perform a “ps -ef” on UNIX you will see many “oninit” process running. Each one is performing a specific task for the database server. Use the UNIX pid column from “ps -ef” to correlate a process to the pid column from “onstat -g sch”. Figure 16 contains example output of this command.

Figure 16: Virtual Processor status: onstat -g sch

```
lester@merlin >onstat -g sch

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:56:46 -- 10656
Kbytes

VP Scheduler Statistics:

  vp         pid         class     semops         busy waits       spins/wait
   1          230          cpu          21                 0                0
   2          231          adm           0                 0                0
   3          232          lio         277                 0                0
   4          233          pio          62                 0                0
   5          234          aio      144794                 0                0
   6          235          msc         756                 0                0
   7          236          aio       64028                 0                0
   8          237          tli           3                 0                0
```

List SQL statement types: onstat -g sql

This is the most interesting of the new options. This option allows you to drill down and see the actual SQL statement that a user is executing. Figure 17 shows an example of listing a summary of all the SQL statements running. Then, by using the session id, you can see details and the actual SQL statements being run. Figure 18 contains an example of this detail.

Figure 17: List all SQL statements: onstat -g sql

```
lester@merlin >onstat -g sql

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:52:02 -- 10656
Kbytes

Sess  SQL            Current             Iso Lock       SQL  ISAM F.E.
Id    Stmt type      Database            Lvl Mode       ERR  ERR  Vers
264   INSERT         ffsdw               NL  Not Wait   -264 0    7.23
```

List SQL statement for a specific user: onstat -g sql sid

Figure 18: SQL statement of a user: onstat -g sid

```
lester@merlin >onstat -g sql 264

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:51:10 -- 10656
Kbytes

Sess  SQL            Current             Iso Lock       SQL  ISAM F.E.
Id    Stmt type      Database            Lvl Mode       ERR  ERR  Vers
264   INSERT         ffsdw               NL  Not Wait   -264 0    7.23

Current SQL statement :
  insert into gjsum select  exp_org,  exp_prog,  bud_obj_code,  job_num,
    sum (exp_amount) from genjournal group by 1, 2, 3, 4

Last parsed SQL statement :
  insert into gjsum select  exp_org,  exp_prog,  bud_obj_code,  job_num,
    sum (exp_amount) from genjournal group by 1, 2, 3, 4
```

This option is very useful in a couple of cases. One is when you do not have access to the SQL code and need to optimize your database tables and indexes. By running this command repeatedly, you can see the SQL statements that are processed. Then, by collecting and examining the SQL, you can determine where to add indexes to improve the performance of the system.

A second use for this option is to debug program transactions. I used this to help a programer debug his program by running "onstat -g sql sid" while he was running his program. I could see error conditions and SQL errors that he was not catching in his program.

List users sessions: onstat -g ses

This option shows additional information about users' sessions, including how much memory each session is using. Figure 19 shows an example. This option can also be used to display detailed information about a session.

Figure 19: List users sessions: onstat -g ses

```
lester@merlin >onstat -g ses

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 7 days 12:57:48 -- 10656
Kbytes

session                                   #RSAM    total      used
id      user     tty      pid   hostname threads  memory     memory
265     informix -        0     -        0        8192       4680
264     lester   4        4249  merlin   1        106496     97840
10      informix -        0     -        0        8192       4680
7       informix -        0     -        0        16384      13144
6       informix -        0     -        0        8192       4680
4       informix -        0     -        0        16384      13144
3       informix -        0     -        0        8192       4680
2       informix -        0     -        0        8192       4680
```

Repeat ONSTAT commands: -r

To continually repeat an ONSTAT command use the "-r # of seconds" option. This is very useful when you need to monitor a situation. The following example displays the status of the logical logs every 10 seconds.

```
onstat -l -r 10
```

Clear ONSTAT shared memory statistics: onstat -z

The OnLine statistics are reset every time OnLine is restarted. To reset all the statistics while OnLine is running, without shutting it down, use the following command:

```
onstat -z
```

This will clear all statistics for ONSTAT and the SMI tables.

ONCHECK - Check and display information about OnLines's disk space

ONCHECK is the tool to check and display information about your dbspaces, blobspaces, chunks, tables, indexes, and disk pages. The purpose of this utility is to insure that your database server disk space has no inconsistencies. I like to think of this as the database version of the UNIX utility 'fsck' which checks file systems, or the DOS utility 'chkdsk' which checks DOS disk space.

ONCHECK operates in two basic modes with two basic options. Figure 20 has a complete list of all the options. The '-c' list of options perform consistency checks and display a limited amount of information unless there is a problem. The '-p' list of options perform the consistency checks and display much more information about what you selected.

When ONCHECK finds a problem it will provide you with an error message to indicate what the problem is. If the problem is a corrupt index, ONCHECK will prompt you to tell it to fix the index. The only problem ONCHECK can fix is corrupt indexes. However, it may be faster to drop and re-create the index using SQL commands than for ONCHECK to fix it.

ONCHECK will place locks on all tables and databases that it needs to access. In some cases it will place an exclusive lock on a database or table and prevent other users from accessing the data. You need to be careful when you run ONCHECK to make sure it will not disrupt your users' work. It is a good idea to run ONCHECK when the OnLine Server is in quiescent mode so it does not conflict with other users. This is especially true when using the '-c' option to check and repair data.

Figure 20: ONCHECK syntax

```
oncheck [-c{check options} ] [-p{display options} ] [-qny ]
        [ { database[:[owner.]table[,fragdbs|#index]]
                | TBLspace number
                | Chunk number } { rowid | page number }
        ]
        -c      Options to check consistency
                r       Reserved Pages
                e       Extents
                c       Database catalogs [database]
                i       Table indexes database[:[owner.]table[#index]]
                I       Table indexes and rowids in index database[:[owner.]table[#index]]
                d       TBLspace data rows including bitmaps database[:[owner.]table[,fragdbs]]
                D       TBLspace data rows including bitmaps, remainder pages and BLOBs
                        database[:[owner.]table[,fragdbs]]

        -p      Options to check and display
                r       Reserved Pages
                e       Extents report
                c       Catalog report [database]
                k       Keys in index database[:[owner.]table[#index]]
                K       Keys and rowids in index database[:[owner.]table[#index]]
                l       Leaf node keys only database[:[owner.]table[#index]]
                L       Leaf node keys and rowids database[:[owner.]table[#index]]
                d       TBLspace data rows database[:[owner.]table[,fragdbs]] [rowid]
                D       TBLspace data rows including bitmaps, remainder pages and BLOBs
                        database[:[owner.]table[,fragdbs]] [page number]
                t       TBLspace report database[:[owner.]table[,fragdbs]]
                T       TBLspace disk utilization report database[:[owner.]table[,fragdbs]]
                p       Dump page for the given database[:[table[,fragdbs]] and rowid |
                        TBLspace and page #]
                P       Dump page for the given chunk num and page num[chunk # and page #]
                B       BLOBspace utilization for given table(s) database[:[owner.]table[,fragdbs]]
        -q      Quiet mode - print only error messages
        -n      Answer NO to all questions
        -y      Answer YES to all questions
```

Checking and displaying the Reserved Pages: oncheck -pr | oncheck -cr

The first 12 pages of the rootdbs contain crucial information OnLine needs to operate. If these pages are damaged, your database server cannot operate. The command to check and print out the Reserved Pages is ‘oncheck -pr’. This is a useful command to run every night or at regular scheduled intervals to make sure everything is all right. Figure 21 shows the first part of the output from ‘oncheck -pr’.

Figure 21: Output from the oncheck -pr command to print Reserved Pages

```
lester@merlin >oncheck -pr

Validating INFORMIX-OnLine reserved pages - PAGE_PZERO

    Identity                       INFORMIX-OnLine Copyright(C) 1986-1995  Infor
    Database system state          0
    Database system flags          0
    Page Size                      2048
    Date/Time created              06/27/97 08:39:23
    Version number of creator      4
    Last modified time stamp       0


Validating INFORMIX-OnLine reserved pages - PAGE_CONFIG
    ROOTNAME                       rootdbs
    ROOTPATH                       /u3/dev/rootdbs1
    ROOTOFFSET                     0
    ROOTSIZE                       500000
    MIRROR                         0
    MIRRORPATH
    MIRROROFFSET                   0
    PHYSDBS                        rootdbs
    PHYSFILE                       2000
    LOGFILES                       12
    LOGSIZE                        1000
```

Problems with the Reserved Pages or any dbspace disk pages can be caused by disk failures, operating systems errors, or server crashes. However, most of the problems I have seen have been created by human error in configuring database servers. Two OnLine servers using the same disk space where the rootdbs offset overlaps can cause one server to overwrite the Reserved Pages of another. One client I worked with had two DBA’s that were not communicating very well. One created and configured an OnLine server using four chunks. The second DBA configured another server and unknowingly used chunk 2 from the first server as the rootdbs. This created a comical situation where when one DBA brought up an OnLine server, the other server would crash.

Another configuration issue is with the operating system. Some versions of UNIX use the first few pages of a disk to store configuration information. If you install your rootdbs using these pages, you will get corrupt Reserved Pages. A client I worked had used these first few pages, and found the Reserved Pages corrupt after every reboot of the computer. Each time the operating system rebooted, it would check each disk for a valid disk label on the first page of the disk. It would not find one, because OnLine had overwritten it with the rootdbs. The operating system would then overwrite part of the rootdbs with a valid disk label. Some Logical Volume managers use the first page to hold configuration information. Most problems with the Reserved Pages can be avoided with careful configuration. Check with your hardware and operating system vendor for information on how it handles the first pages of disk or a Logical Volume.

Checking and displaying the database System Tables: oncheck -cc | oncheck -pc

The System Tables are the key structures which define all the tables, columns, indexes, stored procedures, and constraints for a database. This option checks, or checks and displays, the consistency of these structures. Again, this is another command that should be run on a regular basis to monitor your server. Figure 22 shows part of the output from this command.

Figure 22: Part of the output of the command oncheck -pc to print database datalogs

```
lester@merlin > oncheck -pc stores7
    Database:  stores7
    Owner                       lester
    Date created                07/04/97 17:34:01
TBLspace stores7:informix.systables
    Physical Address            119fa9
    Creation date               07/04/97 17:34:01
    TBLspace Flags              802        Row Locking
                                           TBLspace use 4 bit bit-maps
    Maximum row size            104
    Number of special columns   0
    Number of keys              2
    Number of extents           1
    Current serial value        112
    First extent size           8
    Next extent size            8
    Number of pages allocated   8
    Number of pages used        6
    Number of data pages        3
    Number of rows              44
    Partition partnum           1048736
    Partition lockid            1048736

    Extents
         Logical Page  Physical Page        Size
                    0        119eef            8

    Index information.
        Number of indexes           2
        Data record size            104
        Index record size           2048
        Number of records           44
```

Checking and displaying indexes: oncheck -cI database:table | oncheck -pI

The index structure is one of the key features to fast retrieval of data. In the early days of Informix products, before OnLine, when the database used UNIX files to hold data and indexes, Informix had a utility called "bcheck" to check and fix index structures. I used to run this utility on a weekly basis to make sure all my database indexes were OK. In OnLine, this function is built into the ONCHECK utility. To check all the indexes of a database type:

```
oncheck -cI database_name
```

To check the indexes of a specific table you would type:

```
oncheck -cI database_name:table_name
```

Keep in mind that the larger your table is, the longer this will take to run and it will lock users from updating key values in the table.

If it finds a corrupt index, you will be prompted to fix that index. Enter 'y' and press return to fix the index. On large tables it is faster to drop and create the index using the SQL commands than to use ONCHECK. Fixing indexes will lock the table in exclusive mode. The SQL commands to drop and re-create an index are "drop index index_name" and "create index". Refer to the Informix SQL Syntax manual for the details.

The '-pI' option displays some very useful information about an index. Over time, as data is added and deleted from an index, it will grow by splitting a page to start new pages. Sometimes there is a performance benefit to dropping and rebuilding an index.

Checking and displaying Data Pages: oncheck -cD database:table | oncheck -pD

Data Pages are the pages where your table records are actually stored. This option checks or prints the pages that contain your data records. It is useful to provide a consistency check on your data.

Checking and displaying table space utilization: oncheck -pT database:table

This option checks a table's usage of disk space, and the '-p' print option display some very useful information about a table. See Figure 23 for an example of the output. Some of the key values this displays are:

Number of extents - This shows how fragmented your table is. I like to limit my databases to no more than 1 extent a year. Over 8 extents and you may notice a performance degradation, and depending on the size of your table, at around 180-200 extents, your table will not be able to add any more extents. When this happens, new records cannot be inserted. To fix this problem you must rebuild your table. The best way to accomplish that is to unload it, drop it, re-create it, and reload the data.

Number of pages allocated - This is the number of pages the table has on disk. It will include data pages, free space for new records, and indexes pages.

Number of pages used - This is the number of pages in use. To find out how many free pages you have subtract this value from the number of pages allocated.

Number of data pages - This is the number of pages used to store data records.

Index usage report - This shows how much of the index pages are used and how much is free.

Figure 23: Output from oncheck -pT to print table space utilization

```
lester@merlin >oncheck -pT stores7:items

TBLspace Report for stores7:lester.items

    Physical Address               119fca
    Creation date                  07/04/97 17:34:08
    TBLspace Flags                 801        Page Locking
                                              TBLspace use 4 bit bit-maps
    Maximum row size               18
    Number of special columns      0
    Number of keys                 3
    Number of extents              1
    Current serial value           1
    First extent size              8
    Next extent size               8
    Number of pages allocated      8
    Number of pages used           5
    Number of data pages           1
    Number of rows                 67
    Partition partnum              1048769
    Partition lockid               1048769

    Extents
         Logical Page  Physical Page        Size
                    0        11a039            8

TBLspace Usage Report for stores7:lester.items

    Type                  Pages      Empty  Semi-Full       Full  Very-Full
    ---------------- ---------- ---------- ---------- ---------- ----------
    Free                      3
    Bit-Map                   1
    Index                     3
    Data (Home)               1
                     ----------
    Total Pages               8

    Unused Space Summary

        Unused data slots                                23
        Unused bytes per data page                       18
        Total unused bytes in data pages                 18

    Home Data Page Version Summary
```

```
                  Version                                 Count

                        0 (current)                          1

Index Usage Report for index  104_10 on stores7:lester.items

                    Average    Average
    Level    Total No. Keys Free Bytes
    ----- -------- -------- ----------
        1        1       67       1015
    ----- -------- -------- ----------
    Total        1       67       1015

Index Usage Report for index  104_11 on stores7:lester.items

                    Average    Average
    Level    Total No. Keys Free Bytes
    ----- -------- -------- ----------
        1        1       67       1501
    ----- -------- -------- ----------
    Total        1       67       1501

Index Usage Report for index  104_12 on stores7:lester.items

                    Average    Average
    Level    Total No. Keys Free Bytes
    ----- -------- -------- ----------
        1        1       67       1334
    ----- -------- -------- ----------
    Total        1       67       1334
```

Dumping the contents of a page: oncheck -pp | oncheck -pP | oncheck -pB

This option is useful when debugging. It dumps out the contents of a page from disk.

Checking and displaying extents: oncheck -pe

This option shows how your tables are spread out over chunks. It produces a report by dbspace and check, listing each extent for each table with the starting address and size. In version 5 of OnLine this is the only way to determine how spread out a tables extents were. In version 7 we can use the sysmaster database to get the same information in a more useful manner. In the chapter on the SMI and sysmaster database, I list several scripts that provide this information. However, this option does check the consistency of your extents and still has a very useful purpose. Figure 24 is an example of the output from this command.

Figure 24: Output from oncheck -pe

```
lester@merlin >lester@merlin >oncheck -pe

DBspace Usage Report:  rootdbs             Owner:  informix  Created: 06/27/97

    Chunk: 1   /u3/dev/rootdbs1                      Size     Used     Free
                                                   250000    10337   239663

        Disk usage for Chunk 1                           Start    Length
        -------------------------------------------  --------- ---------
        ROOT DBspace RESERVED Pages                          0        12
        CHUNK FREE LIST PAGE                                12         1
        TBLSPACE TBLSPACE                                   13        50
        PHYSICAL LOG Pages                                  63      1000
        LOGICAL LOG Pages - Log 1                         1063       500
        LOGICAL LOG Pages - Log 2                         1563       500
        LOGICAL LOG Pages - Log 3                         2063       500
        LOGICAL LOG Pages - Log 4                         2563       500
        LOGICAL LOG Pages - Log 5                         3063       500
        LOGICAL LOG Pages - Log 6                         3563       500
        LOGICAL LOG Pages - Log 7                         4063       500
        LOGICAL LOG Pages - Log 8                         4563       500
        LOGICAL LOG Pages - Log 9                         5063       500
        LOGICAL LOG Pages - Log 10                        5563       500
        LOGICAL LOG Pages - Log 11                        6063       500
        LOGICAL LOG Pages - Log 12                        6563       500
        DATABASE TBLSPACE                                 7063         4
        sysmaster:informix.systables                      7067         8
        sysmaster:informix.syscolumns                     7075        16
        sysmaster:informix.sysindexes                     7091         8
        sysmaster:informix.systabauth                     7099         8
        sysmaster:informix.syscolauth                     7107         8
        sysmaster:informix.sysviews                       7115         8
        sysmaster:informix.sysusers                       7123         8
        sysmaster:informix.sysdepend                      7131         8
        sysmaster:informix.syssynonyms                    7139         8
        sysmaster:informix.syssyntable                    7147         8
```

ONMODE - change OnLine operating mode

The ONMODE utility has several key functions: it changes the operating mode of OnLine, shuts OnLine down, allows you to change some configuration parameters on the fly, and provides a means to kill user database connections. Figure 25 provides the complete syntax to the ONMODE command.

Figure 25: ONMODE syntax

```
onmode -abcDdFklMmnpQRrSsuyZz
        -a      <kbytes> Increase shared memory segment size
        -b      <version> Revert OnLine disk structures to an older version
        -c      Perform a checkpoint
        -D      <max PDQ priority allowed> Set max PDQ
        -d      {standard|{primary|secondary <servername>}} set Data Replication server type
        -F      Free unused memory segments
        -k      Shutdown completely
        -l      Force switch to next logical log
        -M      <decision support memory in kbytes> Set size of Decision Support Memory
        -m      Go to multi-user on-line mode from quiescent mode
        -n      Set shared memory buffer cache to non-resident
        -O      Override dbspace down blocking a checkpoint
        -p      <+-#> <class>   Start up or remove virtual processors of class cpu, aio, lio, pio,
                shm, soc, or tli
        -Q      <max # decision support queries> Set max number of Decision Support queries
        -R      Rebuild the /INFORMIXDIR/etc/.infos.DBSERVERNAME file
        -r      Set shared memory buffer cache to resident
        -S      <max # decision support scans> Set max number of Decision Support Scans
        -s      Shutdown to single user (Graceful shutdown)
        -u      Shutdown and kill all attached sessions (Immediate Shutdown)
        -y      Do not require confirmation mode changes
        -Z      <address> heuristically complete specified transaction
        -z      <sid>   Kill specified database session id
```

Shutting down the database server

One of the most common uses of the ONMODE utility is to shutdown the OnLine database server. To immediately shutdown the OnLine server from any mode, type:

```
onmode -ky
```

The '-k' option takes the database server off-line and the 'y' avoids the prompt to confirm your action. This immediately take the server off-line, disconnecting all users. Any user in the middle of a transaction will have their transaction rolled back to the state before they started their transaction. Any work the user was doing will be lost. You must be the user 'root' or 'informix' to perform this function.

Figure 26 shows the error message you will get if you are not logged in as 'informix' or 'root' to shutdown OnLine. It also shows the message you and your users will receive when OnLine has been shutdown and you try to access the database server.

Figure 26: Shutting down OnLine

```
lester@merlin >onmode -ky

Must be a DBSA to run this program
lester@merlin >su informix
Password:
lester@merlin >onmode -ky
lester@merlin >onstat -
shared memory not initialized for INFORMIXSERVER 'train1'
```

When all the electrical power is about to fail, or the computer is shutting down for whatever reason, you don't have time to ask all users to log off the database server. And if you do not shutdown OnLine, it will crash in an inconsistent state, with data in memory buffers that is not been written to disk, and users in the middle of transactions. When OnLine is later restarted in will start a recovery mode to clean up from the crash, but this can take time and there may be problems.

One useful method of invoking this command is to put it in the UNIX shutdown script for your machine. This way when the computer is stopped it will automatically stop OnLine. Check with your UNIX System Administrator on the location of the script. I like to add a call to a separate shell script that uses ONMODE to shutdown the server and ONINIT to start it up, based on a parameter passed to the script. See Figure 27 for an example script that starts and stops multiple OnLine Servers. A good way to test such a startup script is to execute it as root, using the Bourne Shell with none of the Informix environment variables set.

Figure 27: OnLine startup and shutdown script

```
##############################################################################
# Module:       %W%     Date: %D%
# Author:       Lester B. Knutsen             email: lester@access.digex.net
#               Advanced DataTools Corporation
# Discription:    Informix Online startup/Shutodwn script
#     This script is used to start and stop 3 OnLine Servers
#     used for training named: train1, train2, train3
##############################################################################
# Set Global environment variables
##############################################################################
## Set the location of Informix Programs
INFORMIXDIR=/u3/informix7
export INFORMIXDIR

## Add the Informix Programs to your PATH
PATH=$INFORMIXDIR/bin:$PATH:/usr/ccs/bin
export PATH
```

```
############################################################################
# Process and shutdown server
############################################################################
## Set the Database Server
INFORMIXSERVER=train1
export INFORMIXSERVER

## Set the Informix Configuration File
ONCONFIG=onconfig.train1
export ONCONFIG

state=$1
case $state in
      start)
            oninit;
            echo "Informix Server: $INFORMIXSERVER Started";;
      stop)
            onmode -ky;
            echo "Informix Server: $INFORMIXSERVER Shutdown";;
      *)
            echo "usage: ifx.rc start|stop";;
esac

############################################################################
# Process and shutdown server
############################################################################
## Set the Database Server
INFORMIXSERVER=train2
export INFORMIXSERVER

## Set the Informix Configuration File
ONCONFIG=onconfig.train2
export ONCONFIG

state=$1
case $state in
      start)
            oninit;
            echo "Informix Server: $INFORMIXSERVER Started";;
      stop)
            onmode -ky;
            echo "Informix Server: $INFORMIXSERVER Shutdown";;
      *)
            echo "usage: ifx.rc start|stop";;
esac

############################################################################
# Process and shutdown server
############################################################################
## Set the Database Server
INFORMIXSERVER=train3
export INFORMIXSERVER

## Set the Informix Configuration File
ONCONFIG=onconfig.train3
export ONCONFIG

state=$1
case $state in
      start)
            oninit;
            echo "Informix Server: $INFORMIXSERVER Started";;
      stop)
            onmode -ky;
            echo "Informix Server: $INFORMIXSERVER Shutdown";;
      *)
```

```
            echo "usage: ifx.rc start|stop";;
esac
```

Changing OnLine modes

In addition to the ‘onmode -k’ option to shutdown, ONMODE has three other options to change the mode of OnLine. The ‘-k’ option completely shut down the database server and takes it off-line. There are two options that take the database server to quiescent mode. Quiescent mode is like a maintenance mode or single-user mode where you can access OnLine with the utilities but users cannot connect.

The command to gracefully take OnLine to quiescent mode is:

```
onmode -s
```

The command to immediately take OnLine to quiescent mode is:

```
onmode -u
```

The difference between these is that the ‘-s’ option will wait until all users have disconnected before changing modes, and the ‘-u’ option will change modes immediately and kill all connected users.

To return to on-line mode rrom quiescent mode so users can once again access the database server, the command is:

```
onmode -m
```

Forcing a checkpoint

A checkpoint is one of the key events when OnLine syncs shared memory with what is on disk. Several activities depend on the last completed checkpoint. An OnLine archive takes its start date and time from the last checkpoint. You cannot delete a logical log that contains the last checkpoint. To force OnLine to perform a checkpoint, use the following onmode command and option:

```
onmode -c
```

Forcing a switch in the current logical log

Another option to ONMODE allows you to change the current logical log to the next logical log in sequence. This is required if you are going to backup the current logical log or to drop the current logical log. The command and option to change the current logical log is:

```
onmode -l
```

Free unused virtual memory segments

As OnLine runs, it will add additional virtual shared memory segments as needed. Since this operation has some overhead, OnLine does not release unused memory segments, but saves them for future reuse. The ‘onstat -g seq’ command discussed earlier in the chapter shows you the current virtual memory segments. The command to force OnLine to reorganize its virtual memory segments and free unused segments is:

```
onmode -F
```

This operation requires some overhead and will freeze all user processing while OnLine reorganizes and frees this segment. Because of the overhead of free memory and then re-adding it later, this operation should only be done when required. Monitor your virtual memory segments with the command ‘onstat -g seg’. When you notice an increase in the virtual memory segments, and you see that these are no longer being used, then it may be useful to free them with this command. A common occurrence of this is after running large weekly or month-end batch jobs and reports. These type of jobs will often require extra memory that will be used until the next cycle of processing. This is a good opportunity to use this command. Do not repeatedly run this command at short intervals to free memory. The overhead of freeing memory and then re-acquiring it will slow things down.

Killing users' database processes

ONMODE provides an option to kill and abort an individual user's database process. This option is aware of a user's database transaction and will rollback any work that was not committed. Operating system commands to kill a user's process (e.g. the UNIX kill -9 command) are not aware of a user's database connection and may not cleanly rollback their work. This can lead to corruption of tables or indexes. The correct procedure to kill a user's database process is:

1. Identify the user's session id using the ONSTAT command with one of the following three options:
   onstat -u
   onstat -g sql
   onstat -g ses

2. Use the following omode command to terminate the user's session:
   onmode -z session_id

Figure 28 shows an example of identifying the session id for the user 'lester' using 'onstat -u' and killing the session with 'onmode -z' The session id is 190.

Figure 28: Terminating a user's session

```
lester@merlin >onstat -u

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 28 days 11:52:49 -- 10656
Kbytes

Userthreads
address  flags   sessid   user     tty      wait     tout locks nreads
nwrites
a2d0018  ---P--D 1        informix -        0        0    0     114      486
a2d0458  ---P--F 0        informix -        0        0    0     0        5657
a2d0898  ---P--B 8        informix -        0        0    0     2        0
a2d1558  ---P--D 12       informix -        0        0    0     0        2
a2d1998  Y--P--- 190      lester   0        a3d1d50  0    1     18       0
 5 active, 128 total, 17 maximum concurrent

lester@merlin >onmode -z 190
lester@merlin onstat -u

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 28 days 11:53:00 -- 10656
Kbytes

Userthreads
address  flags   sessid   user     tty      wait     tout locks nreads
nwrites
a2d0018  ---P--D 1        informix -        0        0    0     114      486
a2d0458  ---P--F 0        informix -        0        0    0     0        5657
a2d0898  ---P--B 8        informix -        0        0    0     2        0
a2d1558  ---P--D 12       informix -        0        0    0     0        2
 4 active, 128 total, 17 maximum concurrent
```

When you terminate a user's session, their processing may not stop immediately. If they are performing a large query, data may be buffered up on the user's program and they may continue to receive data. Once the buffer is empty, the user will receive an error message indicating the database connection was lost.

Another delay will occur if the user was in the middle of a large update or load in a transaction. OnLine will need to rollback the transaction. The general rule that I use is that if the user was in a transaction for 30 minutes loading data, it will take about 30 minutes to rollback their work. OnLine must delete all the records it has inserted. You must let the rollback complete.

Using ONMODE for configuration changes

ONMODE has several options that allow you to change your configuration while the database server is up and running. This saves changing the ONCONFIG file, shutting down the server, and then restarting it with the new configuration. However, changes made using ONMODE are not written to the ONCONFIG file and will be lost when OnLine is shutdown and restarted.

The following options to ONMODE allow changes:

-a	<kbytes> Increase shared memory virtual segment size
-b	<version> Revert OnLine disk structures to an older version of OnLine (e.g 5, 6)
-d	{standard|{primary|secondary <servername>}} set Data Replication server type
-n	Set shared memory buffer cache to non-resident
-O	Override dbspace down blocking a checkpoint
-p	<+-#> <class> Start up or remove virtual processors of class cpu, aio, lio, pio, shm, soc, or tli
-R	Rebuild the /INFORMIXDIR/etc/.infos.DBSERVERNAME file
-r	Set shared memory buffer cache to resident

Decision support Configuration changes

-D	<max PDQ priority allowed> Set max PDQ
-M	<decision support memory in kbytes> Set size of Decision Support Memory
-Q	<max # decision support queries> Set max number of Decision Support queries
-S	<max # decision support scans> Set max number of Decision Support Scans

ONLOG - debug the logical log

The ONLOG utility allows you to debug transactions using the logical logs. This utility should be used with care as it can display lots of information. The best use of this utility is to research why a user's transaction failed. Figure 29 contains the syntax for this command.

Note: Running ONLOG on the current logical log file (the default) will lock the log file and stop all user processing.

Figure 29: ONLOG syntax

```
onlog [-l] [-q] [-b] [-d <tape device>] [-n <log file number>] [-u <user name>]
        [-t <TBLspace number>] [-x <transaction number>]

        -l      Display maximum information about each log record including hex dump
        -q      Do not display program header
        -b      Display information about logged BLOB pages (-d option only)
        -d      Read from tape device
        -n      Display the specified log(s)
        -u      Display the specified user(s)
        -t      Display the specified TBLspace(s)
        -x      Display the specified transaction(s)
```

ONINIT - initialize and start-up functions

The ONINIT utility starts OnLine and without any other options brings it into on-line mode so users can connect and go to work. This is one of the key commands along with ‘onmode -ky’ to start and stop your database server. See Figure 27 for a script that uses these commands to automatically start or stop OnLine.

However, ONINIT is also one of the most dangerous commands. With a single option (-i) it will initialize you rootdbs, wiping out anything that was there and all your hard work. There may be times you will want to do this but be very careful and make sure all your environment variables are set correctly.

Starting ONLINE

The command to start OnLine is very simple. Just type ‘oninit’, and it will use four environment variables to identify the database server to start. The environment variables are:

INFORMIXDIR - Points to the directory where the Informix products are installed and is used by OnLine as a base directory to locate other files it needs.

PATH - This is used by your operating system to search for executables and must include the directory located in $INFORMIXDIR/bin.

INFORMIXSERVER - This is the name of the OnLine server you wish to start. This name is also located in your ONCONFIG file.

ONCONFIG - This is the name of the configuration file OnLine will use to start the database server. It is located in $INFORMIXDIR/etc.

Once these are set and you are logged in as the user ‘root’ or ‘informix’, type ‘oninit’ to start the server. Figure 27 was an example of a script that will start several OnLine servers. Normally you want your OnLine server to start automatically every time you boot your computer. This script can be called from one of the UNIX startup scripts like ‘/etc/rc.local’ to perform this function. Check with your operating system administrator to find the name of the UNIX startup script that you will need to call this script from.

Figure 30 shows all the options to ONINIT.

Figure 30: ONINIT syntax

```
oninit [-I] [-p] [-s] [-y] [-V] [-v]
          -I     Startup and initialize rootdbs.  This will destroy any existing data in your rootdbs
          -p     Startup and do not delete temp tables during shared memory initialization
          -s     Startup and stay in quiescent mode
          -V     Display version
          -v     Startup in verbose mode.  Many additional messages will display that are          helpful
                 in debugging problems.
```

Initializing the root dbspace

The ‘-I’ option of ONINIT is very powerful and dangerous. It will startup OnLine and initialize your rootdbs. This process is like formatting a dbspace and will destroy all data that is there. However, this option is very handy when you know what you are doing and want to initialize your rootdbs. When I need to configure several database servers it is easier to write a script to do it rather then type all the commands and edit all the options by hand. Figure 35, shows an example script that does this, and uses the next two utilities we will talk about to initialize a rootdbs, set up several dbspaces, and move the logical logs to a separate dbspace.

Verbose Option

Another helpful option is the lower case '-v' for verbose. This displays extra messages as OnLine goes through the different stages on initialization and is very helpful when debugging an installation. Figure 31 shows the output from this option on my training system.

Figure 31: Verbose startup of OnLine with the '-v' option

```
lester@merlin >oninit -v
Reading configuration file '/u3/informix7/etc/onconfig.train1'...succeeded
Creating /etc/.infxdirs ... succeeded
Creating infos file "/u3/informix7/etc/.infos.train1" ...
"/u3/informix7/etc/.conf.train1" ... succeeded
Writing to infos file ... succeeded
Checking config parameters...succeeded
Allocating and attaching to shared memory...succeeded
Creating resident pool 2160 kbytes...succeeded
Creating buffer pool 402 kbytes...succeeded
Initializing rhead structure...succeeded
Initializing ASF ...succeeded
Initializing Dictionary Cache and Stored Procedure Cache...succeeded
Onlining 0 additional cpu vps...succeeded
Onlining 2 IO vps...succeeded
Forking main_loop thread...succeeded
Initialzing DR structures...succeeded
Forking 1 'ipcshm' listener threads...succeeded
Forking 1 'tlitcp' listener threads...succeeded
Starting tracing...succeeded
Initializing 1 flushers...succeeded
Initializing log/checkpoint information...succeeded
Opening primary chunks...succeeded
Opening mirror chunks...succeeded
Initializing dbspaces...succeeded
Validating chunks...succeeded
Forking btree cleaner...succeeded
lester@merlin >Initializing DBSPACETEMP list
Checking database partition index...succeeded
Checking location of physical log...succeeded
Initializing dataskip structure...succeeded
Checking for temporary tables to drop
Forking onmode_mon thread...succeeded
Verbose output complete: mode = 5

lester@merlin >onstat -

INFORMIX-OnLine Version 7.23.UC1   -- On-Line -- Up 00:00:47 -- 10656 Kbytes
```

Figure 32 shows the output to the OnLine message log of a successful startup. There are two key messages you need to look for in the message log when OnLine starts. One is the message 'Physical Recovery Completed: 0 Pages Restored', and the other is 'Logical Recovery Complete'. The zeros for 'Pages Restored' and 'Rolled Back' mean that everything was shutdown cleanly and restarted cleanly with no loss of users' work.

Figure 32: Startup messages in the OnLine Message Log

```
Sat Aug  9 23:52:16 1997

23:52:16  Event alarms enabled.  ALARMPROG = '/u3/informix7/log_full.sh'
23:52:17  DR: DRAUTO is 0 (Off)
23:52:18  INFORMIX-OnLine Initialized -- Shared Memory Initialized.
23:52:18  Physical Recovery Started.
23:52:18  Physical Recovery Complete: 0 Pages Restored.
23:52:18  Logical Recovery Started.
23:52:21  Logical Recovery Complete.
       0 Committed, 0 Rolled Back, 0 Open, 0 Bad Locks

23:52:22  Dataskip is now OFF for all dbspaces
23:52:22  On-Line Mode
23:52:22  Checkpoint Completed:  duration was 0 seconds.
```

ONSPACES - Adding, deleting, and changing mirroring of dbspaces

This utility allows you to add, drop and change the mirroring of dbspaces. It is the equivalent of the menu options in ONMONITOR. If you seldom change your dbspace configuration it is easier to use the menus in ONMONITOR. The ONSPACES utility is very useful if you need to create scripts to change your dbspace configuration. Figure 33 has the syntax for this utility. Figure 35 is an example of a script that uses this utility to configure a database server from scratch.

Figure 33: ONSPACES syntax

```
onspaces  { -a spacename -p pathname -o offset -s size [-m path offset] |
          -c {-d DBspace [-t] | -b BLOBspc -g pagesize} -p pathname -o offset -s size
                    [-m pathoffset]
          -d spacename [-p pathname -o offset] [-y] |
          -f [y] off [DBspace-list] | on [DBspace-list] |
          -m spacename {-p pathname -o offset -m path offset [-y] |
          -f filename} |
          -r spacename [-y] |
          -s spacename -p pathname -o offset {-O | -D} [-y] }
```

-a	Add a chunk to an existing DBspace or BLOBspace
-c	Create a new DBspace or BLOBspace
-d	Drop a DBspace, BLOBspace or chunk
-f	Change dataskip default for specified DBspaces
-m	Add mirroring to an existing DBspace or BLOBspace
-r	Turn mirroring off for a DBspace or BLOBspace
-s	Change the status of a chunk

Note: You can only drop a dbspace if it is completely empty.

ONPARAMS - Change logical and physical log configuration

This utility allows you to add logical logs, drop logical logs, and change the location of the physical log. Figure 34 shows the syntax for this command. This utility is handy because you can do some things with it that cannot be done with ONMONITOR. It allows you to add logical logs of different sizes and locations. One common use of this is after you have set up your server, you will often want to move your logs out of the rootdbs into their own dbspaces. Figure 35 contains an example of a script using this command to move the logical logs to their own dbspace and the physical log to its own dbspace.

Figure 34: ONPARAMS syntax

```
onparams { -a -d DBspace [-s size] | -d -l logid [-y] |-p -s size [-d DBspace] [-y] }
```

-a	Add a logical log
-d	Drop a logical log
-p	Change physical log size and location
-y	Answer YES to all questions

In order to drop a logical log, the log must be backed up and cannot contain the current checkpoint or current logical log.

Figure 35: Script to initialize a database server, add dbspaces, and add logs

```
#############################################################################
# Module:       %W%     Date: %D%
# Author:       Lester B. Knutsen
#               Advanced DataTools Corporation
# Discription:    Script to Creat a training environment Informix OnLine
#           database server
#############################################################################
```

```
############################################################################
# Set Global environment variables
############################################################################

## Set the location of Informix Programs
INFORMIXDIR=/u3/informix7
export INFORMIXDIR

## Add the Informix Programs to your PATH
PATH=$INFORMIXDIR/bin:$PATH:/usr/ccs/bin
export PATH

## Set the Database Server
INFORMIXSERVER=train2
export INFORMIXSERVER

## Set the Informix Configuration File
ONCONFIG=onconfig.train2
export ONCONFIG

############################################################################
# Check that this is the correct ONCONFIG and INFORMIXSERVER
############################################################################

set `grep "^DBSERVERNAME" $INFORMIXDIR/etc/$ONCONFIG`
if [ "$2" != "$INFORMIXSERVER" ]
then
        echo "Invalid INFORMIXSERVER: $INFORMIXSERVER"
        exit
fi
echo "Creating and Initializing INFORMIXSERVER: $INFORMIXSERVER"
echo "Press RETURN to continue"
read ans

############################################################################
# Create the disk devices - this training server uses cooked files
# but you could replace these command with the commands to use raw files.
############################################################################

touch /u3/dev/rootdbs2
touch /u3/dev/logdbs2
touch /u3/dev/rootdbsM2
touch /u3/dev/data2dbs2
touch /u3/dev/tempdbs2

# Set owner to informix - group informix
chown informix:informix /u3/dev/rootdbs2
chown informix:informix /u3/dev/logdbs2
chown informix:informix /u3/dev/rootdbsM2
chown informix:informix /u3/dev/data2dbs2
chown informix:informix /u3/dev/tempdbs2

# Set permissions to read/write owner and group only
chmod 660 /u3/dev/rootdbs2
chmod 660 /u3/dev/logdbs2
chmod 660 /u3/dev/rootdbsM2
chmod 660 /u3/dev/data2dbs2
chmod 660 /u3/dev/tempdbs2
```

```
############################################################################
# Initialize the rootdbs - after this anything that was there is wipped out
############################################################################

oninit -i

## must sleep long enough for the sysmaster database to be created or the
## next step will fail.
sleep 200

# Display the log
onstat -m

## now shutdown to single user mode
onmode -sy

# Display status
onstat -

############################################################################
# Creat the additional Dbspaces
############################################################################

echo "Creating logdbs..."
## Create dbspace for logical logs
onspaces -c -d logdbs -p /u3/dev/logdbs2 -o 0 -s 25000

echo "Creating datadbs..."
## Create dbspace for data
onspaces -c -d datadbs -p /u3/dev/data2dbs2 -o 0 -s 50000

echo "Creating tempdbs..."
## Create dbspace from temp tables
onspaces -c -d tempdbs -t -p /u3/dev/tempdbs2 -o 0 -s 10000

############################################################################
# Create additional logical logs in logsdbs
############################################################################

echo "Creating additional Logical Logs"
onparams -a -d logdbs -s 4000
onparams -a -d logdbs -s 4000
onparams -a -d logdbs -s 4000
onparams -a -d logdbs -s 4000
onparams -a -d logdbs -s 4000
onparams -a -d logdbs -s 4000

echo "Creating archive to activate new Logical Logs"
ontape -s

############################################################################
# Show message log and status
############################################################################

onstat -m

echo "OnLine Configuration complete"

############################################################################
```

ONTAPE - the OnLine backup and restore utility

ONTAPE is the basic utility to backup and restore the whole OnLine server. The backup may be performed while the system is running and while users are accessing and updating data. In addition to your basic backup and restore ONTAPE performs a few other functions:

Backups the logical logs.
Provides a utility to change the logging mode of a database.
Performs a restore to start Data Replication.

OnLine backup strategies and procedures are very important and merit a whole chapter. This is just a quick overview of ONTAPE's syntax and a few options. Figure 36 contains the syntax for ONTAPE.

Figure 36: ONTAPE syntax

```
ontape { -a | -c | -l | -p | -r [-D DBspace_list] | -s [-L archive_level]
          [-A database_list] [-B database_list] [-N database_list] [-U database_list] }
```

-a	Automatic backup of logical logs
-c	Continuous backup of logical logs
-l	Logical restore
-p	Physical restore for Data Replication (HDR)
-r	Full restore DBspaces/BLOBspaces as listed
-s	Archive full system
-A	Set the following database(s) to ansi logging
-B	Set the following database(s) to buffered logging
-N	Set the following database(s) to no logging
-U	Set the following database(s) to unbuffered logging

Limitations of ONTAPE

The basic ONTAPE restore option restores the whole database server. You cannot restore just one table or database. Use ONUNLOAD and ONLOAD to perform database and table level backups.

ONTAPE can only restore to a the same dbspace configuration. The disk layout must match exactly the disk layout when the backup was made.

The ONTAPE backup is binary and can only be restored to a computer which is binary compatible and using the same version of Informix.

Backing up the OnLine server

ONTAPE provides a way for you to backup the whole database server while it is running. ONTAPE will keep track of all changes made during its backup, and during a restore rollback any incompletely backed-up changes. The command to start a backup is:

```
ontape -s
```

ONTAPE uses the parameters in the ONCONFIG file to determine the tape device, block size and tape size. These parameters are:

```
TAPEDEV     /dev/tapedev     # Tape device path
TAPEBLK     16               # Tape block size (Kbytes)
TAPESIZE    1024000          # Maximum amount of data to put on tape (Kbytes)
```

Changing TAPEDEV to /dev/null and performing a backup will reset OnLine's internal parameters without performing an actual backup.

Using ONTAPE requires a dedicated terminal and tape drive during backups only. It will also require an operator to monitor backups and change tapes as needed. For the backups to be used in a restore, the tapes must be labeled carefully and coordinated with Logical Log backup. Figure 37 contains an example of an OnLine backup using ONTAPE.

Figure 37: ONTAPE backup

```
informix@merlin >ontape -s
Please enter the level of archive to be performed (0, 1, or 2) 0

Please mount tape 1 on /dev/rmt/0 and press Return to continue ...
10 percent done.
20 percent done.
30 percent done.

Tape is full ...

Please label this tape as number 1 in the arc tape sequence.
This tape contains the following logical logs:

 9

Please mount tape 2 on /dev/rmt/0 and press Return to continue ...
```

Backing up to disk

Informix does not officially support backing up to disk with ONTAPE but it can be done. Figure 38 contains an example of a shell script that could be used to backup OnLine to a disk file. This could be run by the UNIX Cron facility to automatically backup your server at a set time. However, the backup must be small enough to fit on disk.

Figure 38: Shell script to backup OnLine to disk

```
####################################################################
# Shell script to backup Informix OnLine to disk
####################################################################
## Set Informix environment variables
## change these to match your configuration
INFORMIXDIR=/usr/informix7.1
export INFORMIXDIR
PATH=$INFORMIXDIR/bin:$PATH
export PATH
ONCONFIG=onconfig
export ONCONFIG
INFORMIXSERVER=online1
export INFORMIXSERVER

## Echo message to log file
echo "Archive Informix Online for $INFORMIXSERVER"

## Check for valid backup device, prevents accidentally overwriting
set `grep "^TAPEDEV" $INFORMIXDIR/etc/$ONCONFIG`
if [ "$2" != "/u3/backup/online1.bak" ]
then
        echo "Invalid TAPEDEV $2"
        date
else
        echo "Archive to TAPEDEV $2"
        date
## Start ontape and respond to prompts - the following spacing is key
## There must be a 0 for the level followed by blank line for the
## the response to the prompt.
{
ontape -s <<EOF
0

EOF
} | tail -5
## Only read the last 5 lines to prevent filling up your log when errors
echo "Archive Completed"
fi
```

Restoring an OnLine server

OnLine must be off-line to perform a restore. The restore will wipe out all your current data and configuration. The disk layout must be exactly configured the same as when you created the backup.

Note: Before you begin, write protect your tape. The restore process has a confusing prompt asking "do you want to backup your logical logs?". I know DBA's who have responded yes to this prompt and accidentally wiped out their restore tape.

The command to start a restore is:

```
ontape -r
```

Figure 39 shows the full dialog of prompts during the restore process.

Figure 39: Performing an OnLine restore

```
informix@merlin >ontape -r

Please mount tape 1 on /dev/rmt/0 and press Return to continue ...

Archive Tape Information

Tape type:      Archive Backup Tape
Online version: INFORMIX-OnLine Version 7.13.UC2
Archive date:   Fri Sep 27 17:48:39 1996
User id:        informix
Terminal id:    /dev/pts/0
Archive level:  0
Tape device:    /dev/rmt/0
Tape blocksize (in k): 16
Tape size (in k): 2000000
Tape number in series: 1

Spaces to restore:1 [rootdbs           ]

Archive Information

INFORMIX-OnLine Copyright© 1986-1996  Informix Software, Inc.
Initialization Time       09/03/96 17:31:15
System Page Size          2048
Version                   4
Archive CheckPoint Time   09/27/96 17:48:44

Dbspaces
number   flags    fchunk   nchunks  flags    owner    name
1        1        1        1        N        informix rootdbs

Chunks
chk/dbs offset   size     free     bpages   flags pathname
1   1   0        50000    38641             PO-   /u3/dev/dbspace713

Continue restore? (y/n)y
Do you want to back up the logs? (y/n)n
Restore a level 1 or 2 archive (y/n) n
Do you want to restore log tapes? (y/n)y

Roll forward should start with log number 9

Please mount tape 1 on /dev/rmt/0 and press Return to continue ...
Do you want to restore another log tape? (y/n)n

Program over.
informix@merlin >onstat -

INFORMIX-OnLine Version 7.13.UC2   -- Quiescent -- Up 00:10:35 -- 8976 Kbytes
```

First ONTAPE will display the disk configuration as it was when the backup was performed. This gives you a chance to verify that you have created the correct devices and links.

Next ONTAPE will prompt you if you would like to backup logical logs. This prompt is very confusing if you have not done this a few times. If your system had crashed and OnLine was able to backup additional logs it will help you in the recovery process. Put a NEW tape in the drive and respond yes. Do NOT leave your restore tape in or OnLine may overwrite it. If you do not want to backup any current logs, respond NO.

Now ONTAPE will start the restore. This is no progress report on the restore like there is on the backup. Be patient and wait. If more than one tape is required you will be prompted for it.

After the level 0 tape has been restored ONTAPE will ask if you have a level 1 or 2 backup to restore.

When all backup levels have been restored, ONTAPE will prompt you for logical log tapes to restore. Start with the tape containing the logical log you are prompted for. You cannot skip logical logs or restore them in a different order. If you do not have any logical log backups simply respond NO to these prompts.

Finally, OnLine will start to roll forward, or roll back any transactions necessary. When this is completed the OnLine server will be in quesicent mode ready for you to check.

Backing up logical logs

ONTAPE is also used to backup logical logs. The logical log backup device is controlled by the following parameters in your ONCONFIG file:

```
LTAPEDEV     /dev/tapedev     # Logical Log tape device path (e.g /dev/rmt/0)
LTAPEBLK     16               # Log tape block size (Kbytes)
LTAPESIZE    10240            # Max amount of data to put on log tape (Kbytes)
```

When you do not want to backup Logical Logs to tape, set LTAPEDEV to equal "/dev/null". OnLine understands this and frees the logical logs as soon as they can be reused without a backup. Otherwise, you must backup your logical logs before they can be reused.

Note: When all Logical Logs are full OnLine will halt all processing until you backup the logs. This will stop all user activity.

There are two forms of Logical Log Backup:

1) Continuous Backup (ontape -c) - this runs the ontape process backing up logical logs non-stop until you stop.
2) Automatic Backup (ontape -a) - the ONTAPE process runs backing up all logical logs that need to be backed up. Once all logs have been backed up the process stops.

Issues with continuous backup of logical logs to tape

Continuous backups requires a dedicated terminal or window in which it runs. This is where it will display tape change prompts and expect an operator to respond to these prompts. It also requires a dedicated tape drive and an operator who will monitor the progress of its backups. The operator must carefully label tapes so they can be used in a restore. To stop continuous backups simple enter "Control-C" or the interrupt character on your system.

Anytime the continuous backups is aborted and restarted a new tape must be used or else the old tapes will be overwritten. Each execution of continuous or automatic backups requires a new tape. ONTAPE backups cannot append to the end of the last tape. Continuous backups must also be restarted with a new tape after your system is rebooted.

Issues with automatic manual backup of logical logs to tape

Automatic backup of logs requires a dedicated terminal and tape drive only during the actual backups. This tape drive may be shared with other activities. However, it still requires an operator to monitor the logs, and start a backup before they all become full. It also requires careful labeling of tapes and coordinating with ontape archives. Figure 40 shows the screen display of the automatic backup.

Figure 40: Automatic backup of logical logs

```
Performing automatic backup of logical logs.
Please mount tape and press Return to continue ...
This tape contains the following logical logs:
         11 - 12

Please label this tape as number 1 in the log tape sequence.
Please mount next tape and press Return to continue ...
*** The tape was not changed ***
Please mount next tape and press Return to continue ...
This tape contains the following logical logs:
         12 - 14

Please label this tape as number 2 in the log tape sequence.
Please mount next tape and press Return to continue ...
This tape contains the following logical logs:
         14 - 16

Please label this tape as number 3 in the log tape sequence.
```

Changing logging mode for a database

ONTAPE is also used to change the logging mode of a database. To change a database from no logging to some form of logging requires a backup. The command to perform a backup and change a database from no logging to buffered logging is:

```
ontape -s -B database_name
```

If you want to change the logging mode of a database without performing a complete backup simply change TAPEDEV in your ONCONFIG file to "/dev/null". Then run the ONTAPE command listed above. *Be sure to change TAPEDEV back to its original after you are done.*

ONUNLOAD and ONLOAD - Unloading and loading databases and tables

The ONUNLOAD utility and the corresponding ONLOAD utility provide a way to save whole tables or databases in a binary format to tape or disk. These two utilities copy whole pages from dbspaces and save the results in binary format. They must work together. Only ONLOAD can read and load the results of an ONUNLOAD. If you need to unload data in ASCII or text format use the SQL unload statement of DBEXPORT.

The advantage of these two utilities is that they are fast. They copy whole pages of data including existing indexes structures. This makes it very fast to reload because indexes do not need to be rebuilt. Also, since the output is stored in binary format, there is some security in the data being protected.

There are a few drawbacks to these two utilities:

Data is not portable. It can only be loaded using the same version of OnLine on the same type of machine. Since the data is stored in binary format the operations need to be performed on computers that are binary compatible.

Data is not compressed. Since the data is copied in whole pages, empty data space and empty index structures on a page remain and are reloaded on the target system. SQL unload and load will rebuild the data on pages and rebuild all indexes compressing the data.

The utilities will require an exclusive lock on the table or database being unloaded and loaded.

These utilities are useful in several ways:

They provide fast table level backups.
They provide fast database level backups.
When transferring tables or databases to other systems with different dbspace layouts.
When moving databases or tables from one dbspace to another.

Figure 41 contains the syntax for the onunload command and Figure 42 contains the syntax for the onload command.

Figure 41: ONUNLOAD syntax

```
onunload [-l] [-t <tape device>] [-b <block size>] [-s <tape size>]
                <database> [:[<owner>.]<table>]

        -l      Use logical log tape configuration from ONCONFIG file
        -t      Tape devices overriding TAPDEV in ONCONFIG
        -b      Tape block size overriding size in ONCONFIG
        -s      Tape size overriding size in ONCONFIG
```

Figure 42: ONLOAD syntax

```
onload [-l] [-t <tape device>] [-b <block size>] [-s <tape size>]
        [-d <Dbspace>]
        <database>[:[<owner>.]<table>] [{-i <old indexname> <new indexname>}]
        [{-fd oldDBspname newDBspname}]
        [{-fi indexname oldDBspname newDBspname}]

        -l      Use logical log tape configuration
        -t      Tape devices
        -b      Tape block size
        -s      Tape size
        -d      DBspace name
        -i      Rename index during load
        -fd     Change data fragment dbspace
        -fi     Change index fragment dbspace
```

As an example of using these utilities let's take the steps required to move the items table in the stores7 database from one dbspace to another. These utilities are perfect for this task because they are fast.

First we need to unload the items table. ONUNLOAD requires that the file exist if you are unloading to disk. Our first step is to create an empty file using the UNIX touch command. If you are unloading to tape you can skip this step because the tape device already exists.

```
        touch items.onunload
```

Next we execute the unload:

```
        onunload -t items.onunload stores7:items
```

This will create a binary image of the items table using our file.

Now we need to use SQL and dbaccess to drop the items table. We are piping the SQL statements to dbaccess.

```
        dbaccess stores7 - <<EOF
        drop table items;
        EOF
```

The final step is to reload the items table into a new dbspace. For this example we will load it into a dbspace named itemsdbs.

```
        onload -t items.onunload -d itemsdbs stores7:items
```

This will create the items table in the new dbspace.

Conclusion

These utilities provide a powerful toolkit for the DBA to care for and monitor a database server.

Lester Knutsen
Advanced DataTools Corporation
Phone: 703-256-0267
Web: www.advancedatatools.com
Email: lester@advancedatatools.com