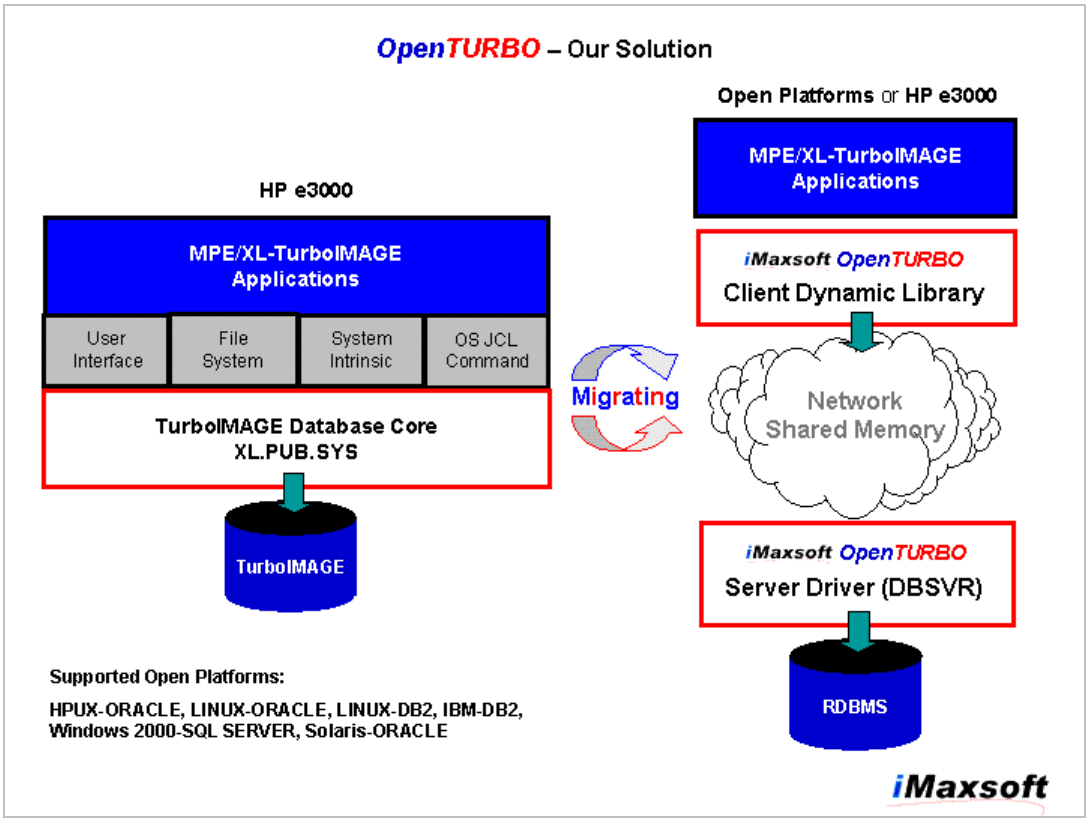
1. Introduction

OpenTURBO is designed and used to seamlessly migrate HP e3000 MPE/XL and TurboIMAGE based applications to an open platform, such as ORACLE on HP-UX, SQL SERVER 2000 on Windows 2000, ORACLE or DB2 on LINUX, and MySQL on UNIX. OpenTURBO reduces migration efforts from years to months, its risk free and speedy methodology can complete and certify 45%-75% of your migration in a few months. OpenTURBO preserves most of your critical business programs intact and keeps your business flow unchanged.

Since HP announced the discontinuation of HP e3000, users are trying to find ways to sustain their HP e3000 hardware and software investment as long as they can; at the same time, they are searching for either a total replacement or a migration path that can lead them into a 'longer life cycle' and 'more open' platform. A significant number of applications have to be migrated, due to 1) it is impossible to find an exact replacement within a reasonable budget, 2) the applications are the products they sell, and 3) it simply cost too much from business aspects, such as business flow re-engineering, training, and hardware and software deployment and replacement.



A full-range of HP e3000 migration consists of many components, depends upon your applications, the weight of each component may variant greatly; but in most cases, database and critical business logic are the two most troublesome items for the migration. These two troublesome items unfortunately are so essential to your core business management and key business decision making, therefore a migration methodology without automated assurance process to your data bank and critical business logic is meaningless. OpenTURBO helps you quickly migrate these two most important items over to an open platform, and our methodology and tools can not only guarantee the migration process, but also assure you the migration results 100%.

Our goal is to completely remove these two most troublesome and worrisome items away from your migration concerns, so you will be able to focus on XML/JAVA GUI development, JCL conversion, Web/Crystal reports creation, Data Warehouse building, IT trainings, and hardware replacement.

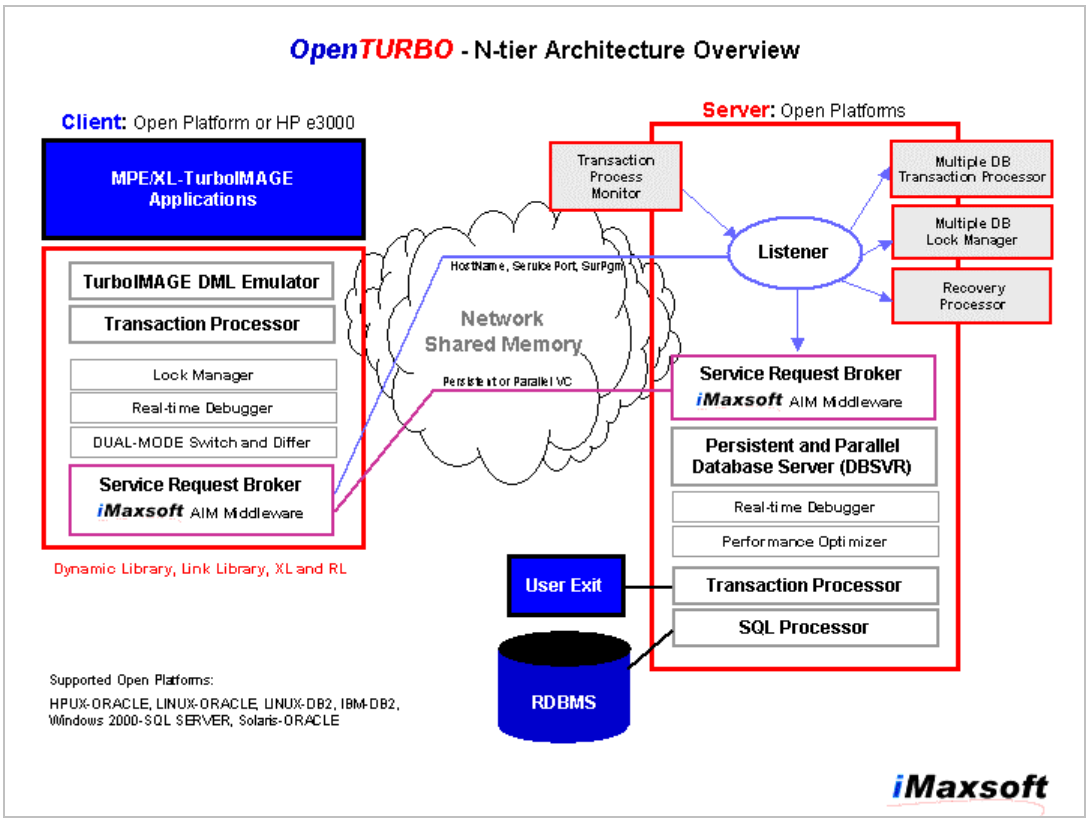
Architecture Overview:

OpenTURBO is a n-tiers client-server architecture, it consists of TurboIMAGE emulator that is used by your existing HP3000 programs to access ORACLE on HP-UX as they were accessing TurboIMAGE on MPE/XL; relational database transformation utilities that is used to replicate and move your entire TurboIMAGE database to ORACLE on HP-UX; and data integrity objects and toolsets that are used to support the co-existence of native relational database and OpenTURBO emulated applications.

TurboIMAGE emulator contains a set of 100% emulated TurboIMAGE API calls and stored in a library which can be a XL on MPE/XL, a shared library on HP-UX, or a dll on Windows 2000. TurboIMAGE emulator libraries are also programming language dependent, for example BBASIC has its own library. You can migrated your MPE/XL TurboIMAGE programs without changing any database related calls and logic, simply link your migrated programs with our TurboIMAGE emulator library and run it. Our emulator will establish necessary connections and contexts for your program to access ORACLE on HP-UX efficiently. On the server, OpenTURBO has a set of processors on the server to serve all requests from TurboIMAGE emulator client and provide responses. OpenTURBO emulator server processors consist of Listener, Lanutil, SQL Server, Lock Manager, Recovery Processor, and Transaction Processor.

OpenTURBO relational database migration utilities transform your entire TurboIMAGE database structure objects, which are user passwords, access classes, keys, paths, sorts, path’s chronological order, unique identifier for each record, and data formatting and conversion, into a relational database ORACLE. OpenTURBO migration utilities can validate that all TurboIMAGE objects and data are properly migrate into ORACLE database, and to ensure that the ORACLE database semantically reflects all attributes of the migrating TurboIMAGE database.

OpenTURBO data integrity objects and toolsets guarantee the integrity of any future TurboIMAGE structure changes and the co-existence of native relational database applications and old MPE/XL and TurboIMAGE applications. OpenTURBO data integrity objects are primary and foreign keys constrains, INSERT, DELETE and UPDATE triggers, row unique identifier sequence number, and lock manager objects. OpenTURBO toolsets allow you to change your ORACLE database and modify your programs in TurboIMAGE mode, this ensures that emulated applications and databases will always behave and act like TurboIMAGE until you replace them in native ORACLE mode.



Frequently Asked Questions:

Question: Why don't we rewrite our applications for HP-UX and ORACLE?

Answer: Your applications will end-up in native HP-UX and ORACLE mode. But, instead of doing it all at once, which may take 5-10 years, we recommend you do it gradually and step by step. Your migration goals should be first move to HP-UX and ORACLE in less than 12 months, then re-create your applications in HP-UX and ORACLE native mode module by module. Any tools or methodology you choose, must allow you to replace one compatibility mode module with a native one and still keep your entire system function as a whole.

Questions: Step by step, what risks are we taking?

Answers: You take no risk at all. OpenTURBO migrates your database and core business logics in a few months, it gives you enough lead-time to evaluate your migration efforts, do your planning and most importantly is that OpenTURBO guarantee the migration results. Here is our OpenTURBO methodology,

Step 1, move your TurboIMAGE database to ORACLE on HP-UX;

Step 2, move your core business programs to ORACLE on HP-UX and run them in emulation mode;

Step 3,

Option 1 of step 3, if you are keeping your HP e3000 box for a while, then your have done your phase 1

migration, because you can run all other parts of HP e3000 applications from HP e3000 and remotely access ORACLE on HP-UX, and gradually making transition of Option 2 tasks;

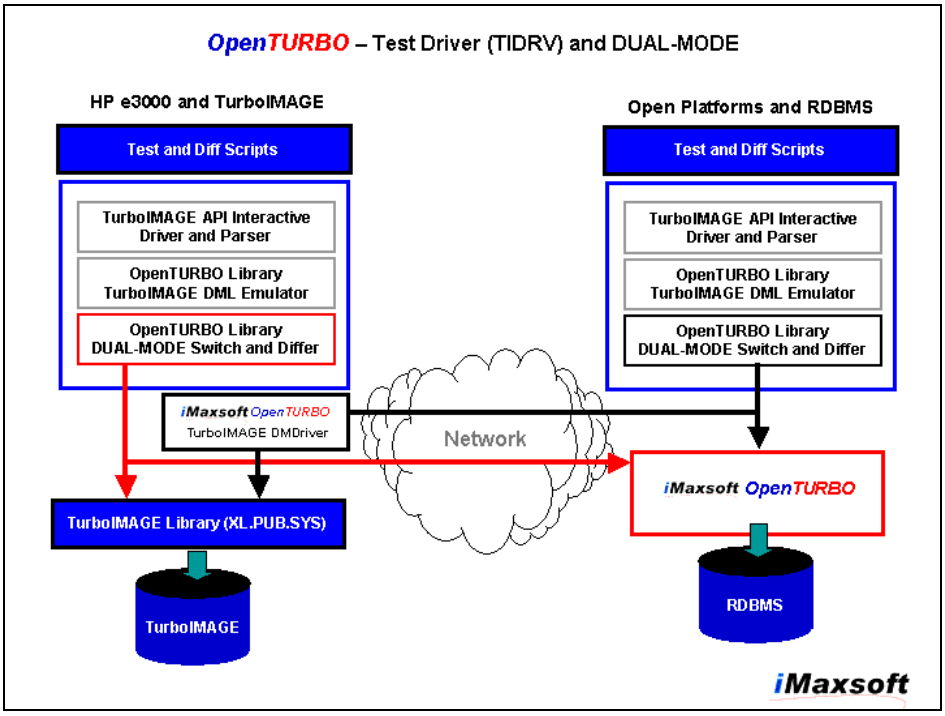
Option 2 of step 3, redevelop your GUI in JAVA, XML or C/S and access ORACLE natively, convert your JCL to HP-UX, replace your Resource Management tools, build your reports and report warehouse;

Step 4, convert your emulated applications into native ORACLE one at a time, at end you will have a complete native migration, the difference is that OpenTURBO will lead your migration efforts step by step in a no risk and low cost methodology.

Question: What? how does OpenTURBO guarantee the migration result?

Answers: 1) We have a set of utilities that can run before, during and after the database migration to assure you that all data are transformed accurately.

2) We have build-in DUAL-MODE feature in OpenTURBO emulator. You can run your old applications on HP e3000 concurrently accessing your local TurboIMAGE database on MPE/XL and your remote migrated ORACLE database on HP-UX, the DUAL-MODE processor will diff your program's database access flow and data buffers, report any differences; if the diff result is NONE by end of the run, you have migrated a 100% identical ORACLE database on HP-UX. You may also run your migrated applications on HP-UX concurrently accessing your local ORACLE database on HP-UX and remote TurboIMAGE database on MPE/XL, the DUAL-MODE processor will diff your program's database access flow and data buffers, report any differences; if the diff result is NONE by end of the run, you have migrated a 100% identical business logic over to ORACLE on HP-UX.



3) OpenTURBO has build-in local and remote debugging facility, it helps you to easily identify the causes of any DUAL-MODE discrepancies, which can be caused by compilers, ORACLE, HP-UX, or OpenTURBO.

Question: ORACLE is too expensive, is it worth to pursue name-brand RDBMS?

Answers: Relational database's core engine is very sophisticated, most name-brand RDBMS has very solid under-line technology. Relational database support and developer tools are also critical to you, name-brand RDBMS vendors tend to have more resources and more complete toolsets.

The name-brand RDBMS vendor normally has better support, in addition to that, you can easily access books from book store and Internet, you can find experienced developers locally, you can access a complete knowledge base for trouble shooting on vendor's Web site, you have more choice of off-the-shelf applications and tools, and you can get trainings almost from anywhere even from community colleges.

In most cases, you don't just pay for the RDBMS core engine, it comes with a lot of tools that can make your future application development quicker and easier.

My experience, your data is the most valuable asset to your business, you cannot afford to use something that is not 100% bullet-proof.

Question: It seems too easy, do you have any proof or success story that you can share with us?

Answers: One of our customer accomplished the followings in 6.8 man weeks:

iMaxsoft Success Story - SEDC

Case – migrates the most sophisticate, the most frequent modify, and the oldest (25+ years) COBOL report program and its corresponding database update program to MF-COBOL on HP-UX ORACLE

Environment and Statistics:
- **HP3000 969/200, 192MM, MPE-XL, TurboIMAGE, COBOLII**
- **HP9000 RP5470 2-way, 1GMM, HP-UX 11.11 ORACLE 9i, MF-COBOL**
- **Report COBOL program size - 20,000+ lines + COPYLIB**
- **Update COBOL program – Item Level LOCK, DBPUT, DBDELETE, DBUPDATE**

1. **COBOL Migration – 6 man weeks (plus filter program and cookbook)**
2. **Database Migration – 8 hours and 19 minutes (end-to-end)**
   2G+, 600,000 Members, and over 1.5m Member History (80% on 969/200)
3. **Time to plug-in *OpenTURBO* for production test – 4 hours**
4. **Test result – reports and updates are 100% matched**
   Performance – 1:59 on MPE/XL TurboIMAGE, 2:01 on HP-UX ORACLE
5. **Total SEDC migration effort - 6.8 man weeks**

iMaxsoft

The first 6 man weeks:

1) migrated the most critical billing program from HP e3000 TurboIMAGE COBOLII to HP-UX MF-COBOL; the main program has over 20,000 lines, it was developed 1975 and is constantly being modified ever since;
2) migrated the core COPYLIB to MF-COBOL on HP-UX;
3) migrated the corresponding billing update program to MF-COBOL on HP-UX;
4) wrote a filter program which can replace MPE/XL system calls with HP-UX ones;
5) developed a cookbook to log compiler centric discrepancies.

The 8 hours and 19 minutes: OpenTURBO transformed its core TurboIMAGE database to ORACLE on HP-UX. The database contains over 600,000 members with 1.5 million members history.

1) On a very slow 969/200, we generated ORACLE schema and loader scripts, and we unloaded data and its path's chronological orders in 6 hours.
2) On a RP5470 2-way HP-UX, we created an entire ORACLE database in 2 hours and 19 minutes.

The 4 minutes:

1) We ran the report program on HP e3000, it took 1 minute and 59 seconds, it created an update file and a report spool file.
2) We ran the migrated report program on HP-UX via OpenTURBO emulator and accessing the transformed ORACLE database, it took 2 minutes and 1 second, it created one identical update file and one identical report spool file.

The last update program:

1) We ran the update program on HP e3000.
2) We ran the migrated update program on HP-UX.
3) We ran QUERY.PUB.SYS in DUAL-MODE to find all updated records and got DIFF_NONE result.

Question: What's their reaction after the 6.8 man weeks?

Answers: DONE!? what we are going to do next, we didn't prepare for this. We need to re-schedule our migration plan, we can officially announce that our applications support HP-UX and ORACLE, and go home with one less worry.

OpenTURBO Main Features:

TurboIMAGE Data Manipulation Language (DML) Emulator:

OpenTURBO provides a set of libraries that 100% emulates the TurboIMAGE API Calls. If your HP e3000 MPE/XL and TurboIMAGE program does nothing but database accessing, data manipulation and business rules calculations, you can move the program straight across to the open platform, recompile it via the similar language compiler, and run it with OpenTURBO libraries.

Dynamic client and server libraries on HP-UX/ORACLE:

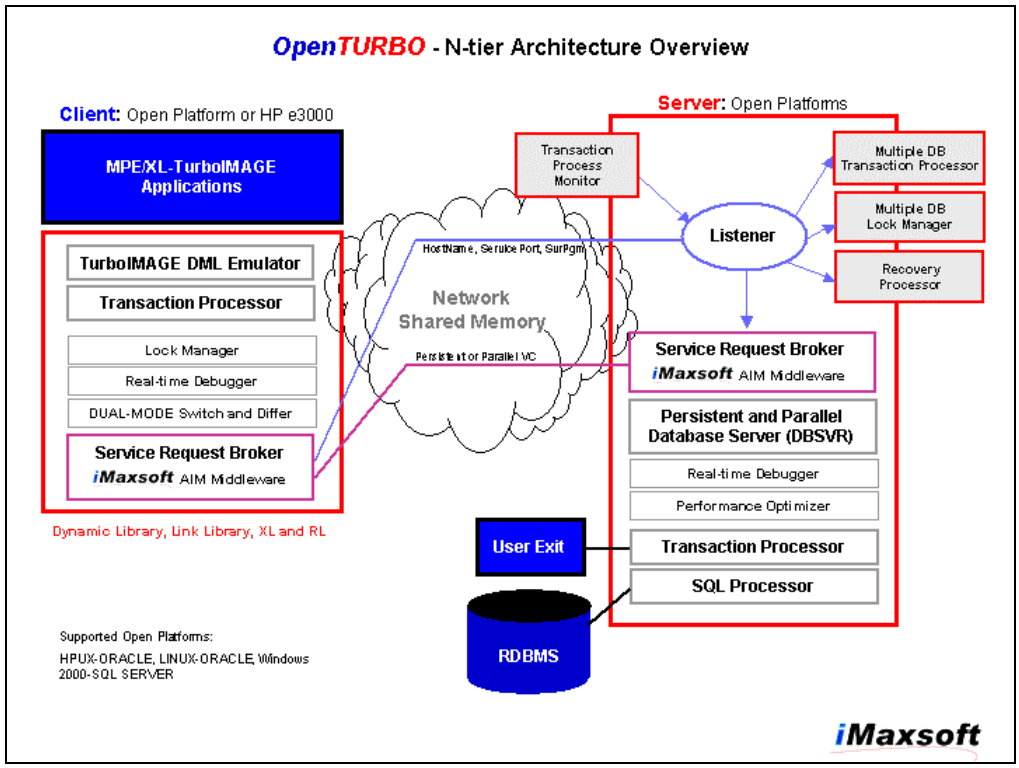
Library Name	Description

liblt	AIM Middleware and Debugging Facility Core Library
libot	OpenTURBO Core Library
libotdbg	OpenTURBO Core Library with Debugger
libsdk	SQL Server Core Library
libsdkc	SQL Client Core Library
libdbutx	DBSVR User Exit Routines Library
libti	TurboIMAGE Test Driver and Mapping Core Library

Dynamic client libraries on MPE/XL for HP-UX/ORACLE:

Library Name	Description
LTXL	AIM Middleware and Debugging Facility Core Library
OTXL	OpenTURBO Core Library
OTXLDBG	OpenTURBO Core Library with Debugger
OTQRY	OpenTURBO Core Library for QUERY.PUB.SYS
OTQRYDBG	OpenTURBO Core Library for QUERY.PUB.SYS with Debugger
DRIVER	Utilities Core Library
TIDRV	TurboIMAGE Test Driver Core Library

Service Request Broker, AIM Middleware, and Database Server (DBSVR):



OpenTURBO Service Request Broker (SRB) consists of Service Request/Reply Library, Listener, and SQL Server.

OpenTURBO/SRB libraries are LTXL on MPE/XL and liblt on HP-UX or open platform. They are a set of routines used to move data across network to different machines and databases, to communicate among applications on the network, and to securely manage and control all traffics on the network.

The listener is a server daemon that accepts OpenTURBO connection requests from network or local loop-back, parses the request and dispatches OpenTURBO database server program (DBSVR).

The database sever program (DBSVR) is the OpenTURBO server program, it can only be spawned by OpenTURBO listener daemon. DBSVR is the core of OpenTURBO, it can be configured in 3 different modes: 1) persistent and direct connect, one DBOPEN per DBSVR, 2) multi-threaded persistent and direct connect, multiple DBOPENs for the same TurboIMAGE database per DBSVR, and 3) parallel and standby, multiple clients to one DBSVR, which is mainly for Web applications and will be available in the next release of OpenTURBO.

You can configure your MPE/XL and TurboIMAGE applications to connect to a TurboIMAGE clone on an open platform, or many TurboIMAGE clones on many different open platforms. Your TurboIMAGE clone can be distributed on a cluster machine or SAN, it can be mirrored and replicated to many other machines in real-time, it is a true relational database running on a native open platform, so it can take advantage of any new and advanced technologies, such as 128bit OS, Application Server, BLOB, JAVA, JDBC, and etc.

Transaction Processor:

OpenTURBO transaction processor supports both single TurboIMAGE database and multiple TurboIMAGE databases transactions (DBXBEGIN, DBXEND, and DBXUNDO).

By default, transaction is at TurboIMAGE call level, any update call (DBPUT, DBUPDATE, DBDELETE) forms a standalone transaction and it cannot be rolled-back. OpenTURBO default transaction handling is exactly like that and no support of rollback.

For DBXBEGIN, DBXEND(COMMIT) and DBXUNDO(ROLLBACK) multiple database transaction, which is transaction across multiple TurboIMAGE databases in a single ORACLE instance, or transaction across multiple TurboIMAGE databases, multiple ORACLE instances and multiple machines. Standard relational database COMMIT and ROLLBCK do **NOT** across process context nor database instance boundaries, hence OpenTURBO has built-in multi-process multi-instance and multi-machine COMMIT/ROLLBACK controller and recovery mechanism, but it is **NOT** 100% data integrity guaranteed, unless you configure OpenTURBO to run with relational database 2PC (2-Phase Commit) option. In the next release of OpenTURBO, we will integrate our transaction processor with relational database's 2PC (2-Phase Commit), therefore OpenTURBO transaction will be database integrity guaranteed regardless of how your databases are partitioned.

Lock Manager:

TurboIMAGE is using abstract lock for concurrency control, and is inter-mixed with database open options. OpenTURBO lock manager provides identical functionality as TurboIMAGE DBOPEN and DBLOCK/DBUNLOCK.

Our lock manager supports database, dataset and row level predicate locks. The client lock controller makes sure there is no lock conflict within the process, and the server lock controller controls and manages all lock requests at single TurboIMAGE database level. In the next release of OpenTURBO,

server lock controller will support multi-TurboIMAGE databases and multi-ORACLE instances locks.

Issues about co-existence of relational versus OpenTURBO applications:

TurboIMAGE is using abstract lock and the lock enforcement is at DBPUT, DBUPDATE, and DBDELETE call level, in order to synchronize relational database locks and OpenTURBO locks are quite challenge and is very expensive, in term of performance cost, to implement an automated equalizer. We highly recommend that you set your standard locking strategies for your future new relational database applications now. Here are some suggestions:

1) Avoid any update conflicts between old OpenTURBO and new native relational database applications; which means un-plug and replace one logical module at a time, don't let ORACLE and OpenTURBO updates the same table at the same time.
2) Apply the same TurboIMAGE abstract locking strategy to your relational database applications; which means a lock object from the same lock must be granted before any ORACLE or OpenTURBO updates.
3) All relational database applications invoke lock object from OpenTURBO abstract lock manager at table update trigger level, and always enforce the abstract and entity locking strategy along with UPDATE INTEGRITY ENFORCEMENT (will be available upon request).

   UPDATE INTEGRITY ENFORCEMENT ensures each updates regardless from native ORACLE or OpenTURBO, a lock object must be granted, the lock object must be issued by the same update process, and the update object is fully covered by the lock object.

   UPDATE INTEGRITY ENFORCEMENT trigger pseudo code (before table INSERT, UPDATE, and DELETE):

```
IF (ORACLE NATIVE) /* Non OpenTURBO Updates */
   IF (OT_LOCK_MANAGER(Update-Object) == EXIST and MINE)
         Proceed-Update
   ELSE
         Update-Object is locked, FAILURE
ELSE /* OpenTURBO Updates */
   Proceed-Update (OpenTURBO Updates)
```

SQL Processor:

Client SQL Processor is in the libraries OTXL and libot. It manages SQL access plan based on the setting in CONFIG file, it synchronizes OpenTURBO and relational database transactions, it controls relational database CURSOR open, keep, switch, and close, it maps item list to column list, it builds the SQL query and update statements, and it converts TurboIMAGE data to/from relational data via OpenTURBO universal data-type formatter which is OS transparent and is always in network byte order during transmission.

Server SQL Processor, DBSVR, is OpenTURBO relational database access core program. It manages all relational database activities and contexts which includes database connection, transaction, cursor and query, update and security.

1) Connection:
   OpenTURBO supports all TurboIMAGE DBOPEN modes from 1 through 8. OpenTURBO guarantees DBOPEN modes control is at physical ORACLE database level even you configure your ORACLE database access is through ORACLE SQLNet.

2) Transaction:

   TurboIMAGE Transaction:

      DBXBEGIN (single or multiple databases)
         DBLOCK(base1)
         DBLOCK(base2)
         DBPUT(base1)
         DBUPDATE(base2)
         DBDELETE(base2)
      DBXEND or DBXUNDO
      DBUNLOCK() – not allowed before DBXEND or DBXUNDO

   OpenTURBO Transaction:

   OpenTURBO always maintains one active ORACLE transaction from the time you open your TurboIMAGE database.

   Single database transaction, DBXBEGIN is mapped to the current active ORACLE transaction that is owned by the persistent DBSVR process, every DBXEND/DBXUNDO is mapped to COMMIT/ROLLBACK which terminates the current transaction and initiates another active transaction immediately afterwards, and this process is autonomy and is data integrity guaranteed.

   Multiple databases transaction, DBXBEGIN is mapped to multiple ORACLE transactions, each transaction is managed independently by its corresponding persistent DBSVR process, every DBXEBD/DBXUNDO is mapped to multiple COMMIT/ROLLBACK which terminates each active ORACLE transaction and initiates active transaction for each DBSVR in a round-robin fashion, if any failures occur during this critical OpenTURBO COMMIT/ROLLBACK process, OpenTURBO transaction controller and recovery mechanism will kick in to undo the damage and re-try the autonomy action if is recoverable; but it has a very remote chance that the recovery process fails and causes data inconsistency in your distributed databases. In the next release of OpenTURBO, you can configure OpenTURBO to run coincide with relational database 2PC (2-Phase Commit) option to ensure your database integrity.

   In a single instance environment, the chance of data inconsistency occurrence is almost **none**, and you should always use relational database logging and recovery features in addition to OpenTURBO transaction recovery processor.

   In a multi-instances and multi-machines environment, the issue is more complex and the recovery is more involved.

** OpenTURBO for ORACLE doesn’t support dirty read. OpenTURBO transaction and isolation level control lays on top of relational database, we support whatever relational database vendor supports in term of dirty read.

3) Data Access - Cursor and Query:

   TurboIMAGE Data Access Methods:

Methods	TurboIMAGE Call
Re-fetch	DBGET Mode = 1
Serial Scan	DBGET Mode = 2 and 3
Direct Fetch	DBGET Mode = 4
Chain Fetch	DBFIND and DBGET Mode = 5 and 6
Calculated Fetch	DBGET Mode = 7
Primary Calculated Fetch	DBGET Mode = 8

TurboIMAGE DBFIND Search Methods:

DBFIND Methods	Description
Mode 1 and 21	KEY = VALUE KEY LIKE VALUE% (B-Tree)
Mode 10	Not Certified by Vendor
Mode 4 and 24 (B-Tree Searches)	KEY = VALUE KEY < VALUE KEY <= VALUE KEY > VALUE KEY >= VALUE KEY IN [VALUE1, VALUE2] KEY LIKE VALUE% KEY LIKE %VALUE%

OpenTURBO Data Access Methods:

TurboIMAGE Call	OpenTURBO Emulation
DBGET Mode = 1	Where IMAXSOFT13_SEQ_NO = current IMAXSOFT13_SEQ_NO
DBGET Mode = 2	Order by IMAXSFOT13_SEQ_NO ASC
DBGET Mode = 3	Order by IMAXSFOT13_SEQ_NO DESC
DBGET Mode = 4	Where IMAXSOFT13_SEQ_NO = input IMAXSOFT13_SEQ_NO
DBFIND & DBGET Mode = 5	Where key = VALUE Order by IMAXSOFT13_PATH_nn ASC
DBFIND & DBGET Mode = 6	Where key = VALUE Order by IMAXSOFT13_PATH_nn DESC
DBGET Mode = 7	Where key = VALUE
DBGET Mode = 8	Same as Mode 7

OpenTURBO Search Methods:

DBFIND Methods	OpenTURBO Emulation
Mode 1 and 21	Where key = VALUE (NUMBER) Where key = ‘VALUE’ (CHAR) Where key like ‘VALUE’ (CHAR)
Mode 10	Not Certified by Vendor
Mode 4 and 24 (B-Tree Searches)	Where key = VALUE Where key < VALUE Where key <= VALUE Where key > VALUE Where key >= VALUE Where key >= VALUE1 and key <= VALUE2 Where key like ‘VALUE%’ Where key like ‘%VALUE%’

OpenTURBO maps Automatic and Manual master datasets to relational B-Tree indexes, so all modes of DBFIND are mapped to standard relational database index scan.

Mode 1: Re-fetch:

The mode 1 of DBGET is simply performing a re-fetch via the current record’s unique key, IMAXSOFT13_SEQ_NO.

Mode 2 and 3: Serial Scan:

Serial scan, during data migration, OpenTURBO will try to preserve your detail dataset overall record's chronological order by assigning a unique number via its raw dataset access order. OpenTURBO assigns a unique number, known as IMAXSOFT13_SEQ_NO, to all records; IMAXSOFT13_SEQ_NO can be either a unique sequence number starts from 1 or the internal TurboIMAGE record number.

TurboIMAGE ***dirty link list*** uses first free first use method for any updates, for those of frequently updated (DBDELETE and DBPUT) detail datasets, whose overall record's chronological order is not accurate at all. But, OpenTURBO only cares the record access order of serial scan not its true chronological order. Internally, OpenTURBO uses the unique sequence number to emulate TurboIMAGE serial scan access order by sorting IMAXSOFT13_SEQ_NO in ascending order for DBGET mode 2, and descending order for DBGET mode 3.

Note 1: the access orders from TurboIMAGE and ORACLE are always identical right after the migration.
Note 2: if you do have TurboIMAGE and ORACLE co-existence requirement; any detail dataset updates after the migration, TurboIMAGE uses dirty link list first free first use algorithm, whereas OpenTURBO uses chronological sequence number; therefore, their serial scan access orders will be no longer in sync from that point on forward.
Note 3: OpenTURBO maintains chronological order for all records in all tables.

Mode 4: Direct Fetch

The mode 4 of DBGET is performing a direct record fetch via a given record number, which is the IMAXSOFT13_SEQ_NO in OpenTURBO environment.

Mode 5 and 6: Cursor Fetch or Chain Get

Chain fetch, OpenTURBO preserves and migrates detail dataset path's chronological order by assigning a sequence number (IMAXSFOT13_PATH_nn) in order to emulate a TurboIMAGE path's forward link list. Path with sort key, OpenTURBO generates WHERE search condition based upon DBFIND argument and sets ORDER BY to 'sort key + all fields after the sort key ascend' for mode 5 DBGET and 'sort key + all fields after the sort key descend' for mode 6 DBGET. Path without sort key, OpenTURBO sets ORDER BY to OpenTURBO's path sequence number, such as 'IMAXSOFT13_PATH_nn ascend' for path-nn in mode 5 DBGET and 'IMAXSOFT13_PATH_nn descend' for path-nn in mode 6 DBGET.

Due to,

i. the chain get is the most frequently used access method,
ii. the inter-changeable forward and backward chain get is allowed in the same data access path,
iii. in most of relational database, a cursor is always forward direction and doesn't return the length (row count) of a cursor,
iv. the cursor resources are expensive and limited in a relational database,
v. overall OpenTURBO performance consideration,

so, OpenTURBO Cursor Manager assures you that critical relational database recourses are managed efficiently, such as CURSOR and TRANSACTION. OpenTURBO Performance Optimizer streamlines data access plans and enhances SQL queries performance. TurboIMAGE DBFIND and DBGET are translated to SQL CURSOR, SQL FTECH, and SQL SELECT, according to your application data access methods and patterns, memory pool utilization rates, and ORACLE access statistics; hence, OpenTURBO translation engine is the key to your application performance. Proper dataset level performance tuning is required for performance improvement and throughput enhancements.

OpenTURBO Cursor Manager and Performance Optimizer configurable conditions:

Condition 1:	The maximum length of a dataset's paths is predictable, see Path Analysis Report by running otANALYZER on HP-UX, and DBFIND mode 1 with no wildcard.
Recommendation:	OpenTURBO loads the entire cursor pointers into memory, therefore OpenTURBO handles all cursor movements in memory, which is very efficient but the price is paid at the Cursor Pointer BULK FETCH (DBFIND). In this case, any subsequent DBGET mode 5 or 6 is simply a unique index scan fetch, which is very efficient.
Condition 2:	Applications performs only one direction DBGET only, never mix DBGET Mode 5 and Mode 6 in the same data access path.
Recommendation:	OpenTURBO can take advantage of Cursor Manager bulk-fetch and pre-fetch features to reduce I/O and speed-up data transport.
Condition 3:	Applications ignores status[5-6] chain count, status[7-8] backward pointer, and status[9-10] forward pointer.
Recommendation:	OpenTURBO can take advantage of Cursor Manager bulk-fetch, pre-fetch and no-cursor-movement features to reduce I/O and speed-up data transport significantly.

Mode 7 and 8: Calculated Get (HASH)

Since hash index is not used in OpenTURBO, both mode 7 and 8 of DBGET are index scan via 'key = search value' condition.

4) Updates:

DBPUT is mapped to ORACLE INSERT:

1. You may only insert Manual and Detail tables.
2. The Automatic table is automatically maintained by OpenTURBO.
3. You may not insert Detail row that has no corresponding key in Manual table.
4. All unique sequence number is maintained by OpenTURBO in chronological order automatically.

5. All Detail paths are maintained by OpenTURBO automatically.

DBUPDATE is mapped to ORACLE UPDATE:

1. You may update any columns in Detail tables.
2. You may only update non-key columns in Manual tables.
3. You may not update Automatic tables at all.
4. CIUPDATE or NO CIUPDATE are the same to OpenTURBO, we implement it simply for syntactic emulation purpose. OpenTURBO maintains path related adjustments with or without CIUDPATE, which is identical to TurboIMAGE CIUPDATE path handling.

DBDELETE is mapped to ORACLE DELETE:

1. You may not delete a Manual entry, unless its foreign constrains are all eliminated.
2. The Automatic entry is not deleted automatically, you need to run otCLEAN for clean-up. The Automatic tables have no value to OpenTURBO, since all Automatic datasets have already mapped to ORACLE indexes for each Detail table. In most cases, Automatic tables are not migrated, hence OpenTURBO does not spend extra performance cost for deleting entry from Automatic table when the entry has no foreign constrains attached to it.

All rules for INSERT, DELETE and UPDATE, such as triggers, primary and foreign constrains, unique constrain and path control, are also applied to native ORACLE applications.

Proper locks must be provided before updates, refer to HP TurboIMAGE/XL Database Management System Reference Manual.

5) Security:

Semicolon is mapped to the root user of the specific TurboIMAGE database, in TurboIMAGE, it is the creator of the database.

All other classes are mapped to users who have specific access authorities to columns and tables.

Internally, OpenTURBO uses SYNONYM name-space feature for connecting users, database objects, and securities at TurboIMAGE database level, which is a logical database of an ORACLE instance.

6) Universal Data-Type Formatter:

OpenTURBO is using network byte order for data transport, which is OS independent.

OpenTURBO converts all data into BINARY mode in a half-word (16-bit) array, just like TurboIMAGE, regardless it is character X, integer I or float R.

OpenTURBO supports all TurboIMAGE data types, I, J, E, R, P, Z, X, U, and K. OpenTURBO converts HP real R into IEEE float E during migration. OpenTURBO handles zoned decimal in both standard and non-standard positive number, which is that the last positive sign byte can be 0 through 9 or {, A through I.

OpenTURBO also supports NLS at both database and emulator levels.

7) DBSVR Modes:

OpenTURBO DBSVR has 3 modes, Dynamic Query Mode, Dynamic Query Multi-threaded Mode, and SQL Pre-process Mode. Each mode has its specific purpose, the Dynamic Query mode is the standard mode, which is used in most of the applications at all time. The Dynamic Query Multi-threaded mode is used for Web applications, and the SQL Pre-process mode is only for super high volume transactions applications and is customizable by customers.

Recovery Processor:

OpenTURBO provides 3 levels of recovery for abnormal process termination,

1. At client process level; any abnormal termination of the client process which is your application program, the OpenTURBO recovery process is triggered automatically to release lock objects, to close open cursors and to disconnect databases.
2. At DBSVR sever process level; any abnormal client process termination, the OpenTURBO recovery process is triggered to terminate client related dangling DBSVR processes, and to re-claim all client and server processes related database control resources.
3. At LISTENER process level; any core abortion of DBSVR server process, the OpenTURBO recovery process is triggered to clean-up dangling server processes and to re-claim all client and server processes related database control resources.

OpenTURBO transaction level recovery: in a multi-database transaction, if any failure occurs during COMMIT or ROOLBACK, OpenTURBO will triggered recovery process to first undo and reset the recovery state back to the first internal COMMIT or ROLLBACK, then re-try the whole COMMIT or ROLLBACK process, the number of re-tries is configurable.

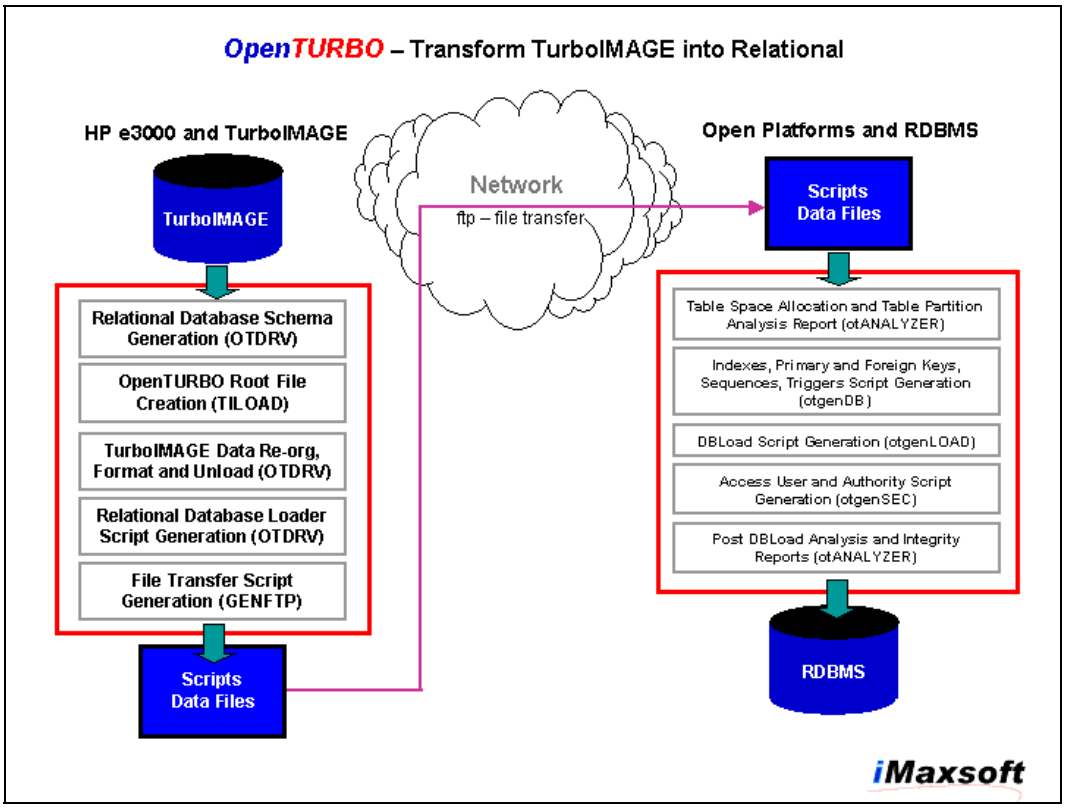
Real-time TurboIMAGE to RDBMS Data Replication – DOOR:

If you need to maintain both TurboIMAGE and ORACLE databases in-sync at all time, and you want to have an exact mirror of TurboIMAGE in ORACLE on HP-UX, DOOR is the solution. A lot of customers, using DOOR as an interim solution to keep an exact mirrored ORACLE on HP-UX during the lengthy migration process, un-plug it when the application is completely migrated.

DOOR intercepts all TurboIMAGE updates, whether is from TurboIMAGE logging files or from TurboIMAGE API calls, transports and replicates changes to ORACLE on HP-UX in real-time mode.

In DOOR, the OpenTURBO unique sequence number is mapped to TurboIMAGE internal record number, so, for serial scan DBGET mode 2 and 3, the data access orders are always in-sync.

OpenTURBO Utilities: transform TurboIMAGE database into ORACLE on HP-UX or other open platforms, and maintain and manage on-going OpenTURBO database structure changes in ORACLE.



ALLBASE/SQL to ORACLE or other RDBMS (per request):

1. SQLGENX - generates ORACLE database from ALLBASE/SQL.
2. FASTLOAD - loads data from ALLBASE/SQL into ORACLE.
3. PSQLX - scans pre-processor code to identify and to flag any code change requirements.

HP e3000 Database Structure Management Utilities:

OTDRV	- Generates ORACLE schema, and sets mapping and data conversion rules.
TILOAD	- Creates OpenTURBO TI Root File, verifies OpenTURBO TI Root File version and re-generates TurboIMAGE schema from OpenTURBO TI Root File.
tiCHG	- Finds changes from old and new TI Root Files, generates OpenTURBO Database Structure Change commands for tiSQLCHG program.

HP-UX Database Structure Management Utilities:

TILOAD	- Verifies OpenTURBO TI Root File version and re-generates TurboIMAGE schema from OpenTURBO TI Root File.
tiSQLCHG	- Converts tiCHG OpenTURBO Database Structure Change commands into ORACLE SQL ALTER statements and performs physical database structure changes.
otgenDB	- Creates additional ORACLE specific database objects on top of standard OpenTURBO relational database creation script.
otgenLOAD	- Enhances ORACLE loader script for performance.

otgenSEC - Generates ORACLE user and access securities based upon TurboIMAGE passwords and access classes.

otCLEAN - Cleans-up Automatic table dangling entries that have no foreign constrains to any Detail tables.

HP e3000 Rapid Data Unload and Transform Utilities:

OTDRV - Unloads data, paths, and unique record identifier from TurboIMAGE datasets into flat files, generates ORACLE loader script and sets data conversion rules.

GENFTP - Generates ftp script for transferring files from HP e3000 to HP-UX.

QUERY.PUB.SYS (Referred to TurboIMAGE/QUERY Manual) – HP e3000 only.

OpenTURBO Real-time Debugger:

1. Programmatically:

   DBCONTROL Mode 88 – turns on or off a specific OpenTURBO debugging level. OpenTURBO supports 32 debugging levels starting from 0 through 31.

   ```
   LTDBG0   - OpenTURBO Internal Core ERROR
   LTDBG1   - OpenTURBO Core Library Call Trace
   LTDBG2   - OpenTURBO Reserved Words
   LTDBG3   - OpenTURBO Error Messages
   LTDBG4   - OpenTURBO Emulator Call Trace
   LTDBG5   - OpenTURBO SDK Call Trace and CURSOR POOL Size
   LTDBG6   - OpenTURBO DUAL MODE Diff Results
   LTDBG7   - OpenTURBO Transaction Performance Trace
   LTDBG13  - TurboIMAGE Call Flow Trace

   LTDBG17  – Network Traffic Dump in Hex and Text formats
   LTDBG18  – Socket Information
   LTDBG19  – Net/IPC Information

   LTDBG27  – Dynamic SQL Statement Preparation Trace
   LTDBG28  – SQL Statement Execution Error
   LTDBG29  – MALLOC, CALLOC and FREE Tracing
   ```

   DBCONTROL Mode 88 – set remote DBSVR debugging levels.
   DBCONTROL Mode 89 – set remote DBSVR debugging output file name.

2. Setup from Client Process: the following setup is only for your-program and output to your tty.

   ```
   EXPORT LTDBG6=1
   EXPORT LTDBGOUT=-
   ./your-program
   ```

3. Setup from Server LISTENER Process: the following setup will trigger all server DBSVR processes share the same debugging output file /users/lee/tmp/2002-07-16.dbg with same debugging levels, 4, 17, 18, 19, 27, 28, and 29.

   ```
   EXPORT LTDBG4=1
   EXPORT LTDBG17=1
   EXPORT LTDBG18=1
   EXPORT LTDBG19=1
   ```

```
EXPORT LTDBG27=1
EXPORT LTDBG28=1
EXPORT LTDBG29=1
EXPORT LTDBGOUT=/users/lee/tmp/2002-07-16.dbg
./listner OTB
```

HP e3000 DUAL-MODE Differ: option to validate database migration.

Your can turn on internal DUAL-MODE option from OpenTURBO HP3000 emulator library by setting OT_DUALMODE = ON in the CONFIG file, and you must set the followings environment variables for logging test results:

```
SETVAR LTDBG6 1
SETVAR LTDBGOUT “difffile.group.account”
RUN yourpgm
```

HP-UX DUAL-MODE Differ: option to validate application migration.

DMDRV is the HP3000 DUAL-MODE driver program that connects to your HP9000 programs for calling TurboIMAGE native APIs remotely.

On HP3000, you must stream the listener job first, JLISTNER.PUB.IMAXSFOT, which spawn child process DMDRV to handle all remote TurboIMAGE calls from your HP9000 program.

JLISTNER File

```
!job listener,mgr.imaxsoft
!COMMENT
!COMMENT ************************************************
!COMMENT *  IMAXSOFT LISTENER - for DUAL-MODE from HPUX   *
!COMMENT ************************************************
!COMMENT
!file hosts.net.sys=hosts.pub.imaxsoft
!file services.net.sys=services.pub.imaxsoft
!purge DMDIFF.LEE > $NULL
!build DMDIFF.LEE;rec=-80,,f,ascii;disc=100000;msg
!setvar ltdbg6 1
!setvar ltdbgout "DMDIFF.LEE"
!run listner.bin.imaxsoft;info="DBA"
!eoj
```

On HP9000, you must set the followings in the CONFIG file:

```
. . .

OT_DUALMODE          = ON
TI_DUALMODE_HOST     = 207.92.64.66
TI_DUALMODE_SERVICE  = 32600
TI_DUALMODE_PGM      = DMDRV.BIN.IMAXSOFT
. . .
```

HP e3000 and HP-UX OpenTURBO Test Driver (TIDRV):

TIDRV is designed to allow you to verify all TurboIMAGE API calls interactively without writing programs. You may create a set of API calls and test it repeatedly, you may direct TIDRV in TurboIMAGE mode only, in ORACLE mode only, or in TurboIMAGE and ORACLE dual-mode. You may direct TIDRV to get input from a file and output to a file or from/to terminal. You may turn on local client debugger to view internal OpenTURBO procedure call traces and data dump, or turn on remote server debugger to view

internal DBSVR SQL statements and before and after data conversion dump.

TIDRV supports it own commands along with TurboIMAGE API call commands, it is the best and most efficient tool that we use to conduct OpenTURBO functional, system, integration and regression tests.

TIDRV is also used for performance benchmarking, its build-in language can easily setup a test suite inter-mixed with different OpenTURBO performance optimizer options and run it for a number of pre-defined cycles for emulating OLTP as well as BATCH transactions.

HP-UX ORACLE Integrity Analyzer (otANALYZER): produces reports for path analysis, tablespace allocation estimates and database migration validation.

OpenTURBO Listener and Transaction Process Monitor:

LISTNER - OpenTURBO client-server daemon, it must be running at all time for listening your application DBOPEN connection request and spawning DBSVR server process for each corresponding client process.

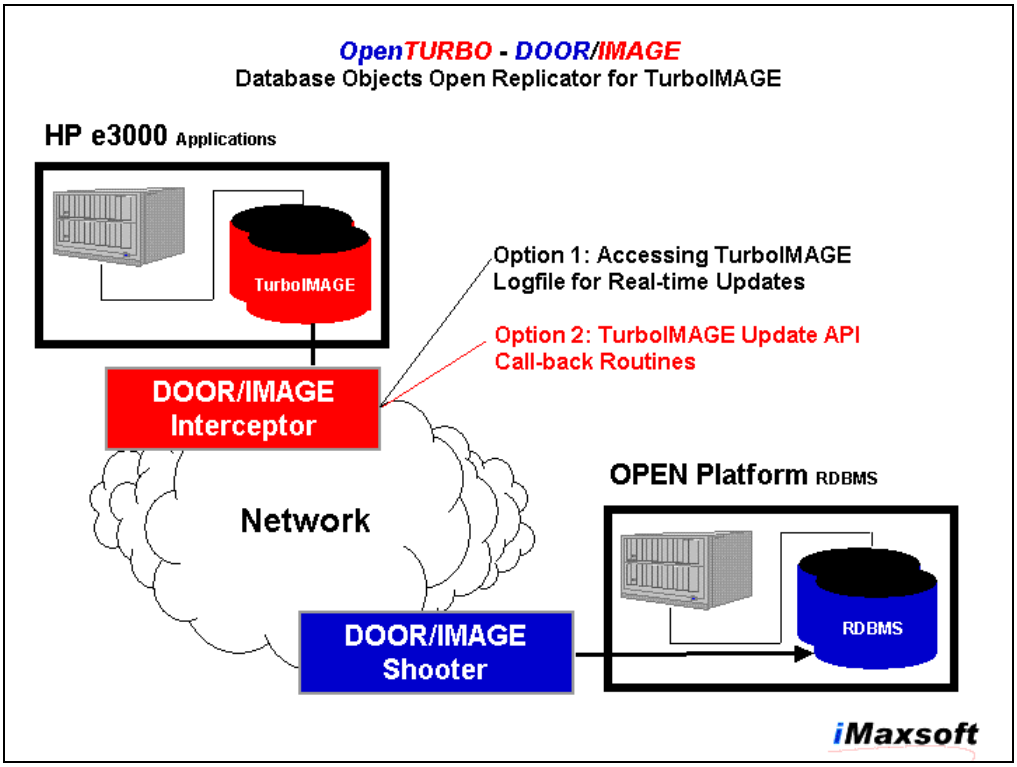
LANUTIL - Tracks listener's activities remotely. You use it to verify who and when a specific user login and use it to gracefully shut down a listener daemon.

Remote TurboIMAGE Access:

OpenTURBO TurboIMAGE Remote Access server – DMDRV.BIN.IMAXSOFT. This program it triggered by apps that is running on NON HP e3000 and is accessing TurboIMAGE remotely via OpenTURBO. DMDRV is a persistent server, it maps to one corresponding DBOPEN() only, and it supports HP and INTEL byte-order conversion.

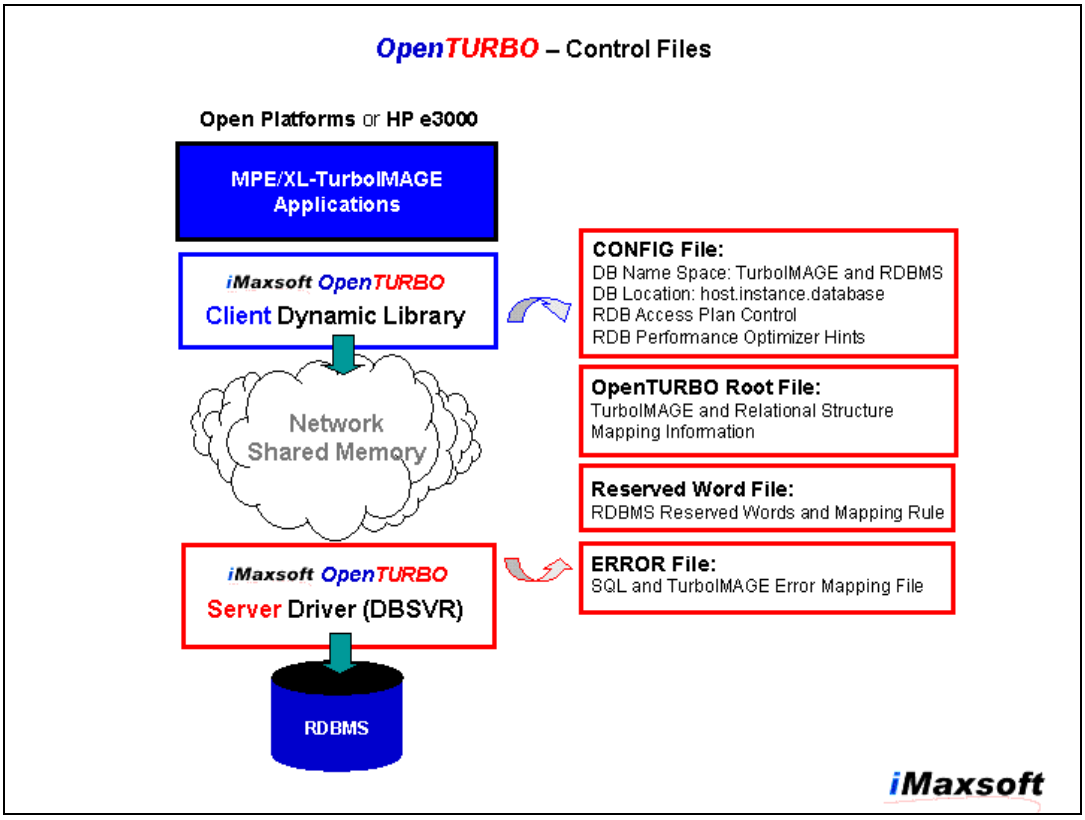
DOOR:

```
DOORMAP    - DOOR Replication Map File Generation
INTERCOT   - DOOR Interceptor, it intercepts TurboIMAGE updates
             either from TurboIMAGE logging files or directly
             from TurboIMAGE API calls, puts data into a queue
             for DOOR Shooter process.
SHOOTOT    - DOOR Shooter – it gets data from DOOR queue and
             transports them to one or many target relational
             databases on the Network.
```



OpenTURBO Configuration and Control files: OpenTURBO Emulator uses four major control files to resolve database names, to set database and dataset access plans, to get TurboIMAGE native database structure and its emulation rules, and to load RDBMS specific reserve words and mapped messages for run-time usage.

OpenTURBO TurboIMAGE Internal (TI) Root-File contains the native TurboIMAGE root file information as well as OpenTURBO emulation rules.



OpenTURBO Emulator CONFIG File:

This is OpenTURBO emulator's client control file, it contains all TurboIMAGE databases attributes, such as OS type, RDBMS type, database location, access methods, and update options, at your enterprise level. It is the most critical file, it should be kept in a secured directory and it should be one and only one for your entire enterprise databases. It provides an overall road map of all your databases at enterprise level. Your need to replicate this file to all client machines for appropriate database accessing for OpenTURBO emulator as well as utilities.

You may use environment variable OT_CONFIG to redirect the CONFIG file; such as, ***export OT_CONFIG=/users/lee/conf1***, then OpenTURBO will use /users/lee/conf1 as the CONFIG file instead of default file CONFIG. One HP3000, you may use either ***:setvar OT_CONFIG 'conf1'*** or ***:file config=conf1*** to redirect the CONFIG file.

```
//    Denotes for comment lines
{}    Begin and End of a Database Definition Block
[]    Begin and End of a Dataset Definition Block
```

Database Level Configuration:

- DBNAME.GROUP.ACCOUNT, DBNAME.GROUP and DBNAME point to the same physical TurboIMAGE database.
- OT_ROOT_FILE points to the root file created by TILOAD. No environment variable is allowed in the file name, such as $HOME/db/DBNAMEti.ti gets error 'TIFile cannot be found'.
- OT_RESERVE_WORD_FILE points to the file that contains relational database reserve words and conversion suffix.
- OT_ERROR_FILE points to the file that contains the TurboIMAGE errors and messages for OpenTURBO.
- OT_HOST points to the machine where your ORACLE instance is logical (SQLNET) or physical located.
- OT_SERVICE directs OpenTURBO to the appropriate listener daemon on the OT_HOST.
- OT_DB_RDBMS specifies the combination of OS and RDBMS, value 515 is ORACLE on HP-UX, and 712 is ORACLE on INTEL-LINUX.
- OT_RDB_LOGON contains TurboIMAGE Class 64 and Password ';' equivalent relational database logon, the format is user/password, and the password is in its encrypted format, use ENCRYPT to generate your encrypted password.
- OT_SDK_SERVER_PRG contains the OpenTURBO Core Server Program name, /opt/imaxsoft/otb/lbin/dbsvr.
- OT_CIUPDATE denotes whether UPDATE Critical Item option is OT_CIUPDATE denotes whether UPDATE Critical Item option is allowed or not; this option is NULL, if you didn't migrate your detail set paths and associated automatic sets.
  To be able to use OT_CIUPDATE=ON correctly,
  1) you need to unload your detail dataset with option –c (OTDRV),
  2) if DOOR is used, you need to generate mapfile with option –c (DOORMAP),
  3) if DOOR is used, you need to start shooter with option –c (SHOOTOT),
  4) set OT_CHRONOLOGICAL = ON for the entire TurboIMAGE database or specific detail dataset.

  The OT_CIUPDATE = ON and OT_CHRONOLOGICAL = ON sets IMAXSOFT_PATH_nn = -888 in DBUPDATE, which triggers ORACLE update trigger to perform Automatic set maintenance procedure.
- OT_IMAGEMODE = ON, enables OpenTURBO to access TurboIMAGE in native mode. If your apps run on HP-UX, then you need to specify TI_DUALMODE_HOST, TI_DUALMODE_SERVICE and TI_DUALMODE_PRG for remote TurboIMAGE access.

The option is also used in conjunction with OT_DUALMODE for specifying the primary database.

- OT_DUALMODE enables OpenTURBO concurrent dual-access of both TurboIMAGE and ORACLE and reports discrepancy.

OS	OT_IMAGEMODE	OT_DUALMODE	Comment
MPE/XL	ON	OFF	TurboIMAGE only
MPE/XL	-	ON	ORACLE primary TurboIMAGE secondary
MPE/XL	OFF	OFF	ORACLE only
HP-UX	ON	OFF	TurboIMAGE only
HP-UX	-	ON	ORACLE primary TurboIMAGE secondary
HP-UX	OFF	OFF	ORACLE only

- TI_DUALMODE_HOST, TI_DUALMODE_SERVICE, and TI_DUALMODE_PGM are used to connect to TurboIMAGE on HP/3000 from your HP-UX migrated programs. You must start the Listener on the HP/3000 before running your program on the HP-UX in DUAL-MODE.
- OT_LOCKWAIT_CYCLE controls the number of wait cycles, when your application uses unconditional LOCK option, each wail cycle is 2 seconds and default cycle is wait forever which is equivalent to OT_LOCKWAIT_CYCLE = 0.
- OT_TRX_THRESHOLD controls the SQL transaction run-time trace when you use ltxldbg (HP e3000) or libotdbg (HP-UX) libraries and trigger OpenTURBO debugging facility via 'SETVAR LTDBG7 1' or 'EXPORT LTDBG7=1'. Default threshold is 10 seconds. It traps any transaction that takes longer than threshold limit and reports it for performance analysis.

Dataset Level Configuration:

- OT_DETAILSETNAME specifies the name of a detail dataset, @ for all detail datasets.
- OT_IGNORE_CHAINSTATUS = ON, OpenTURBO ignores the OT_BULKCHAINGET. OpenTURBO will allow either forward (mode 5) or backward (mode 6) chain get without tracking chain count and forward and backward pointers. If one of your application does only one direction chain get and does not use status words 5-10, the you should use this option to significant improve its through-put and performance.
- OT_BULKCHAINGET = ON, OpenTURBO will trigger bulk fetch and get as many rows as they can to fit internal buffer, whereas in the standard mode, OpenTURBO gets one row at a time;
  OT_BULKCHAINGET option supports one direction chain get only, either DBGET Mode 5 forward or DBGET Mode 6 backward, but not both. Ignored, if OT_IGNORE_CHAINSTATUS = ON.
- OT_CHRONOLOGICAL = ON, OpenTURBO will sort IMAXSOFT13_PATH_nn in ascending order for DBGET Mode 5 and in descending order for DBGET Mode 6.
  In the DBPUT, this option will force OpenTURBO to maintain the chronological order of each IMAXSOFT13_PATH_nn. CIUPDATE is applicable only for DBUPDATE, whereas the OT_CHRONOLOGICAL is for DBPUT, DBUPDATE and DBDELETE.
- OT_IGNORE_DBPUTSTATUS = ON, OpenTURBO returns no record number after successful DBPUT, this is strictly for performance.
- OT_USE_IMAGERECNUM = ON, this is for DUAL-MODE only. OpenTURBO will DBPUT to TurboIMAGE first, then use the returned record number for ORACLE INSERT statement. This is to assure an exact mirror of TurboIMAGE and ORACLE.

```
// Ver:  A.02.01
// For:  SYSTEM LEVEL CONFIGURATION FILE for OpenTURBO
//       This is used by HP/3000 Client
// Date: 2003/06/21
//
// 515 is for HPUX and ORACLE
// 771 is for INTEL-LINUX and ORACLE
//
// OT_TI_DBNAME contains the fully qualified TurboIMAGE database name
// You may have as many ALIAS as you want, avoid making the same alias
// for different TurboIMAGE databases in the same CONFIG file, the first
// found alias is used in all cases.
//
OT_TI_DBNAME = DBNAME.GROUP.ACCOUNT
DBNAME.GROUP.ACCOUNT {
DBNAME.GROUP {
DBNAME {
   OT_IMAGEMODE          = OFF
   OT_ROOT_FILE          = DBNAMEti.ti
   OT_RESERVE_WORD_FILE  = RESERVE.ORACLE
   OT_ERROR_FILE         = OTERROR.ORACLE
   OT_HOST               = 207.92.64.8
   OT_SERVICE            = 32600
   OT_OS_RDBMS           = 515
   OT_RDB_LOGON          = DBANME)GROUP_ACCOUNT/YWIVHZ
   OT_SDK_SERVER_PGM     = /users/lee/lbin/dbsvrA02
   OT_CIUPDATE           = ON
   OT_DUALMODE           = OFF
   TI_DUALMODE_HOST      = 207.92.64.66
   TI_DUALMODE_SERVICE   = 32602
   TI_DUALMODE_PGM       = DMDRV.BIN.IMAXSOFT
   OT_LOCKWAIT_CYCLE     = 60
   OT_TRX_THRESHOLD      = 1
//
// Database level global setting:
//
// OT_HOST
// OT_SERVICE
// OT_OS_RDBMS
// OT_RDB_LOGON
// OT_SDK_SERVER_PRGM
//    The above 5 are for HP-UX and ORACLE server, or where you start OpenTURBO
//    listener. You must specify correct value for OT_OS_RDBMS, 515 for HP-UX
//    ORACLE and 712 for INTEL-LINUX and ORACLE.
//    The OT_RDB_LOGON is the ORACLE logon, you may append SQLNet string to
//    the end of logon. The password must be encrypted via our ENCRYPT program.
//
// OT_CIUPDATE:
//    OT_CIUPDATE = ON: enable TurboIMAGE Critical Item Update option.
//
// OT_DUALMODE
// TI_DUALMODE_HOST
// TI_DUALMODE_SERVICE
// TI_DUALMODE_PGM
//    The above 4 are used to control DUALMODE execution. The OT_DUALMODE is
//    used in conjunction with OT_IMAGEMODE, see details below.
//    The TI_DUALMODE_HOST, TI_DUALMODE_SERVICE, and TI_DUALMODE_PGM are used
//    only when you try to access TurboIMAGE remotely, for both OT_IMAGEMODE
//    and OT_DUALMODE. To be able to run correctly, you must start listener
//    from the HP3000 where the TurboIMAGE databases are.
//
// OT_IMAGEMODE:
//    OT_IMAGEMODE = ON: OpenTURBO accesses the specific database in TurboIMAGE
//    native mode. OpenTURBO ignores all other settings but TI_DUALMODE_HOST,
//    TI_DUALMODE_SERVICE and TI_DUALMODE_PGM, if you intend to access
//    TurboIMAGE database remotely from HP-UX or LINUX.
//
// OT_DUALMODE:
//    OT_DUALMODE = ON - OpenTURBO will perform TurboIMAGE calls in
//    dual mode. OpenTURBO diffs data and status[1]-status[10] for READ/WRITE.
//
```

```
// OS      OT_IMAGEMODE OT_DUALMODE Comment
// ======  ============ =========== ============================================
// MPE/XL       ON          OFF      TurboIMAGE is prime in single mode
// MPE/XL        -           ON      Remote ORACLE is prime and TurboIMAGE is
//                                   secondary
// MPE/XL      OFF          OFF      Remote ORACLE is prime and single mode
// ------------------------------------------------------------------------------
// HP-UX        ON          OFF      Remote TurboIMAGE is prime and single mode
// HP-UX         -           ON      ORACLE is prime and remote TurboIMAGE is
//                                   secondary
// HP-UX       OFF          OFF      ORACLE is prime in single mode
// ======  ============ =========== ============================================
//
// OT_LOCKWAIT_CYCLE:
//    Specify the number of retries allowed for unconditional lock request
//    (mode 1, 3, and 5). The default is 0, wait forever.
//
// Dataset Access Control:
//
// MODE 5 and MODE 6 Chain Get Controls:
// Use @ for the entire database, or special setting for each dataset
//
// OT_IGNORE_CHAINSTATUS:
//    OT_IGNORE_CHAINSTATUS = ON: OpenTURBO ignores the OT_BULKCHAINSET
//    settings. OpenTURBO will perform either forward (MODE 5) or backward
//    (MODE 6) chain get without tracking chain_count, forward and backward
//    pointers. (It is the most efficient option, but status 3-4, 5-6,
//    7-8, and 9-10 are all 0.)
//
// OT_BULKCHAINGET:
//    OT_BULKCHAINGET = ON: Get as many rows as it can fit in a 30000 bytes
//    buffer - ONE DIRECTION ONLY, MODE 5 forward only, MODE 6 backward only.
//    OT_BULKCHAINGET = OFF: Get one row at a time, you may go forward and
//    backward by using the status 7-8 and 9-10 backward and forward record
//    pointers (100% emulates DBGET())
//
// OT_IGNORE_DBPUTSTATUS:
//    OT_IGNORE_DBPUTSTATUS = ON: OpenTURBO returns no record number after
//    successful DBPUT. This is a performance tuning factor.
//
// OT_USE_IMAGERECNUM:
//    OT_USE_IMAGERECNUM = ON: This is for DUAL-MODE only. OpenTURBO will
//    perform TurboIMAGE DBPUT first and then use the returned RecNum for
//    ORACLE INSERT statement. This is to ensure that the ORALCE is exact
//    mirror of TurboIMAGE database.
//
// OT_CHRONOLOGICAL:
//    OT_CHRONOLOGICAL = ON - OpenTURBO will sort the path via OpenTURBO
//    internal chronological sequence number IMAXSOFT13_PATH_nn; ASC for
//    MODE 5, DESC for MODE 6.
//
//    OpenTURBO will maintain the chronological order for each path in the
//    dataset that has OT_CHRONOLOGICAL = ON.
//
//    OT_CIUPDATE = ON - OpenTURBO will set IMAXSOFT13_PATH_nn = -888
//    to inform OpenTURBO Server program to trigger the appropriate
//    UPDATE procedure. The UPDATE triggers are variant depending upon
//    whether the Automatic datasets are used or not. If the CIUPDATE is
//    not enabled, then no key-item is allowed to change, so there is
//    no need to reset chronological order during DBUPDATE.
//
//    Preservation of Detail Dataset Path Chronological Order:
//       1) unload your TurboIMAGE with '-c' option (OTDRV.BIN.IMAXSOFT)
//       2) generate your DOOR mapfile with '-c' option (DOORMAP.BIN.IMAXSOFT)
//       3) start DOOR shooter with '-c' option (SHOOTOT.PUB.DOOR in command
//          file SHOOTCMD.PUB.DOOR)
//       4) OT_CIUDPATE      = ON (database level)
//       5) OT_CHRONOLOGICAL = ON (dataset  level)
//
OT_DETAILSETNAME = @ [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_IGNORE_DBPUTSTATUS  = ON
   OT_USE_IMAGERECNUM     = ON
```

```
   OT_CHRONOLOGICAL       = ON
]
//
// For Detail Dataset CUSTOMER-SET Only
//
OT_DETAILSETNAME = CUSTOMER-SET [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
   OT_CHRONOLOGICAL       = ON
]
}
```

OpenTURBO RESERVED WORD File:

This file is used by emulator library as well as some of the utilities; for emulator, it is identified in the CONFIG file; for utilities, it can be identified via program run-time option, or use file $_OTB_ROOT/conf/RESERVE.ORACLE on HP-UX, or use file that is located via file equation command :FILE RESERVE.ORACLE= on HP3000.

This is an OpenTURBO internal file, if there is no suffix= specified in the file, default _IMS is used. All reserve words are **NOT** case sensitive. You can add or delete reserve words, but is ***not*** recommended, if you have to do so, you must re-do OpenTURBO process from the beginning to re-synchronize your database object name and data, which includes database structure generation, database creation, and data re-loading.

/* .. */ Denotes for comment lines
suffix = Is attached to the end of each reserve word

```
/************************************************************/
/* RESERVED WORDS (ORACLE)         IMAXSOFT Corp. 2002-03-09 */
/************************************************************/
/* Suffix that is attached to the end of each reserved word  */
/* The length of suffix is from 1 to 20 characters           */
/************************************************************/
suffix=_IMS
all
allocate
alter
analyze

. . .
. . .

whenever
where
with
work
year
zone
modify
```

OpenTURBO OTERROR MESSAGE File:

This file is used by emulator only, it is identified in the CONFIG file.

This is an OpenTURBO internal file, you can only change the message text portion and you may not change the error number nor adding or deleting any entries from the file. // denotes for comment lines.

```
// OpenTURBO Error Messages (TurboIMAGE Emulation)
// Date: 2002/04/04
0     = SUCCESSFUL EXECUTION - NO ERROR
-1    = NO SUCH DATABASE
-11   = BAD DATABASE NAME OR PRECEDING BLANKS MISSING
-12   = DATABASE MUST BE IN LOGON GROUP AND ACCOUNT
-13   = NOT ALLOWED; MUST BE CREATOR OF ROOT FILE OR DATABASE
-21   = BAD PASSWORD

. . .
. . .

60    = DATABASE ACCESS DISABLED
61    = PROCESS HAS THE DATABASE OPEN 63 TIMES; NO MORE ALLOWED
69    = BAD DATABASE
```

TurboIMAGE ROOTFILE Internal Structure:

TurboIMAGE ROOTFILE Internal Structure is discussed in great details in TurboIMAGE Handbook and some other publications, here, we simply document those data structures that are directly accessed by OpenTURBO.

TurboIMAGE ROOTFILE (-400):

ROOTFILE Labels:

Label Number	Content	Size
0	ROOTFILE INFORMATION	256 Bytes
1	PASSWORD TABLE	256 Bytes
2	PASSWORD TABLE	256 Bytes
3	ITEM R/W TABLE	256 Bytes
. . .		
M	ITEM R/W TABLE	256 Bytes
M+1	SET R/W TABLE	256 Bytes
. . .		
N	SET R/W TABLE	256 Bytes

LABLE 0: Database General Information

short(2B)	Content
0	Root file condition: JB – Virgin. The DB has not been created yet. FW – OK. RM – Modified deferred. The DB is being modified. MC – The DB is being created. ME – The DB is being erased. IL – ILR recovery in progress. IE – ILR enable in progress. ID – ILR disable in progress. CN – Conversion by DBCONV was in progress and can not be continued. CA – Conversion by DBCONV was in progress and can be continued. MV – DB file move is in progress.
1	Creation date
2-3	Creation time
4	EVEROPEN
5	Cold load ID
6	USERCOUNT
7	DBG dst number
8-11	LOGID – log ID for transaction logging
12-15	LOGPASS – log ID password
16	Database flags: Bit Map Description Default ------- ----------- ------- 0 Recovery NO=0 1 LOGGING NO=0 2 ACCESS YES=1 3 DUMPING NO=0 4 OUTPUT DEFER NO=0 5-6 SYSTEM ACCESS (None=3,R=1,RW=0,0=N/A) R/W=0 7 ILR NO=0 8 ROLLBACK NO=0 9 RESERVED 10 DIRTY FLAG (DB has been modified, but not DBSTORED yet) YES=1 11 DBRECOV RESTART NO=0 12 SWQ NO=0 13-15 RESERVED

17	DBSTORE date
18-19	DBSTORE time
20	Buffer spec count
21	ILR log creation date
22-23	ILR log creation time
24	Last log access date
25-26	Last log access time
27	Previous rollback date
28-29	Previous rollback time
30	Rollback date
31-32	Rollback time
33	Reserved
34	Language ID
35-42	Language mnemonic
43	Number of data set that has been processed by DBCONV
44-63	Reserved
64-67	Database maintenance word
68-127	Buffer specifications 1 Byte: buffer for 1 user 2 Byte: buffer for 2 user . . . 120 Byte: buffer for 120 user

The date format is: bit 0-6 for year, 7-15 for day of year.
The time format is: bit 0-7 for hour, 8-15 for minutes, 16-23 for seconds, 24-31 for tenth of seconds.

LABLE 1 and 2: Password and User Class

Short(2B)	LABLE 1
0-3	Password for user class 0
4-7	Password for user class 1
. . .	
124-127	Password for user class 31
Short(2B)	**LABLE 2**
0-3	Password for user class 32
4-7	Password for user class 33
. . .	
124-127	Password for user class 63

LABLE 3 - M: Item R/W Bit Map

Short(2B)	Content
0-7	Item 1 READ/WRITE bit map
. . .	
. . .	Item m READ/WRITE bit map

Number of Item R/W Bit Map labels is M=((Number_Of_Items*8))/128, and the Max Size is (1023*8)/128 = 64 labels. (Max number of items per TurboIMAGE database is 1023)

LABLE M+1 – N: Set R/W Bit Map

Short(2B)	Content
0-7	Set 1 READ/WRITE bit map
. . .	
. . .	Set n READ/WRITE bit map

Number of Set R/W Bit Map labels is N=((Number_Of_Sets*8))/128, and the Max Size is (199*8)/128 = 13 labels. (Max number of sets per TurboIMAGE database is 199)

Both Item and Set R/W Bit Map has the same user class mapping format, the first 4 short (8 bytes) is for READ, the next 4 short (8 bytes) is for WRITE, and each bit is mapped to it corresponding user class from 0-63.

ROOTFILE Records

Label Number	Content	Size
0	DATABASE GLOBAL INFORMATION	256 Bytes
1	ITEM MAP and SET MAP	256 Bytes
2	ITEM TABLE	256 Bytes
. . .		
M	ITEM TABLE	256 Bytes
M+1	SET TABLE	256 Bytes
. . .		
N	SET TABLE	256 Bytes
N+1	DATA SET CONTROL BLOCKS (DSCB)	256 Bytes
. . .		
P	DATA SET CONTROL BLOCKS (DSCB)	256 Bytes
P+1	DEVICE CLASS TABLE	256 Bytes
. . .		

Record 0: Database General Information

Short(2B)	Content
0	DBSTATUS, first 8 bit contains 'C' for TurboIMAGE Version, then second 8 bit contains 0x02.
1-4	DBNAME
5	Trailer area length
6	The largest block of all data sets in the database, BLOCK LENGTH = BIT MAP SHORT_LENGTH + Media Record Length * Blocking Factor = (Blocking Factor+15)/16 + Media Record Length * Blocking Factor
7-8	Root file length
9	Number of items
10	Number of sets
11	Record number of ITEM_TABLE (in the root file)
12	Record number of SET_TABLE (in the root file)
13	Record number of DSCB (in the root file)
14	Record number of Device_Class_Table (in the root file)
15	DBG flags – 1: Information can fit in DBG 0: Information can not fit in DBG
16-19	Reserved
20	Number of data sets opened.
21	Max number of data sets that can be opened.
22	Restart Date
23-24	Restart Time
25-28	DBCONV Restart File Name
29-32	DBCONV Restart File Group
33-36	DBCONV Restart File Account
37-127	Reserved

Record 1:

Short(2B)	Content
0-32	ITEM MAP
32-63	Reserved
64-95	SET MAP
96-127	Reserved

These maps are used by DBOPEN for speedy access to information in the ITEM_TABLE and SET_TABLE.

Record 2-M: Item Attributes

Short(2B)	Content
0-7	Item-1 Name
8	Item-1 No of Synonym; internal item name search uses hashing, this number contains the number of items

	whose name has the same hashed value.
9	Reserved
10	First Byte: item type, I,J,K,E,R,X,U,Z,P Second Byte: item count for compound item
11	First Byte: item length (single entry length for compound item) Second Byte: unused
12-23	Item-2 Attributes
24-35	Item-3 Attributes
. . .	
(M-11)-M	Item-m Attributes

The max size of Item Attributes Table is 12*1023=12276 (short).

Record (M+1)-N: Set Attributes

Short(2B)	Content
0-7	Set-1 Name
8	First Byte: Number of Synonym; internal set name search uses hashing, this number contains the number of sets whose name has the same hashed value. Second Byte: unused
9	First Byte: unused Second Byte: set type, A,M,D
10-11	DSCB (Data Set Control Block) pointer in the root file which is the short offset from the RECORD_0.
12-23	Set-2 Attributes
24-35	Set-3 Attributes
. . .	
(N-11)-N	Set-n Attributes

The max size of Set Attributes Table is 12*199=2388 (short).

Data Set Control Block (DSCB) – General Layout

# of shorts	DSCB Content
30	SET-1 Global Area: Capacity, Block Length, Media Record Length, ..
FiledCNT*2+1	SET-1 Record Definition Table: Item Numbers List Item Displacements List Media Record Length
PathCNT*3	SET-1 Path Table: Search Item, Sort Item, ..
. . .	SET-2 DSCB
. . .	SET-3 DSCB
. . .	SET-n DSCB

Data Set Control Block (DSCB) – Global Area

Short(2B)	Content
0-1	Data set capacity
2	Block length including the bit map overhead
3	Media record length including the paths pointer overhead
4	Entry length
5	First Byte: blocking factor Second Byte: path count
6	Field count
7	Bit 0: TRUE, the key is hashed Bit 1-15: Primary path Master, field number of the search item Detail, path number of the primary path
8	Path table pointer, short offset relative to its DSCB, the path table contains an entry for each path defined. The pointer points to the 0th entry in

	the table, if you want to get to the first entry in the table, you should increment the pointer by the length of the entry (3 shorts per entry).
9-10	Logical end of file
11-12	Max number of records in the set
13-29	unused

Data Set Control Block (DSCB) – Record Definition Table

Short(2B)	ITEM NUMBERS List
0	Item Number of the first field
1	Item Number of the second field
. . .	
N	Item Number of the last field
Short(2B)	ITEM DISPLACEMENTS List
0	Short offset to the first field
1	Short offset to the second field
. . .	
N	Short offset to the last field
Short(2B)	MEDIA RECORD LENGTH
0	Set media record length in shorts

Data Set Control Block (DSCB) – Path Table

Short(2B)	ITEM NUMBERS List
0-2	Path-1: Master: Byte Description ---- ---------- 1-2 item number of the search item in the related detail set 3-4 item number of the sort item in the related detail set 5 set number of the related detail set 6 path number of the corresponding path in the related detail set Detail: Byte Description ---- ---------- 1-2 field number of the search item 3-4 filed number of the sort item 5 set number of the related master set 6 path number of the corresponding path in the related master set
3-5	Path-2 Attributes
. . .	
(N-2)-N	Path-n Attributes (n<=16)

Record (N+1)-P: Device Class Table

Short(2B)	Content
0-3	Device Class Name 1
4-7	Device Class Name 2
. . .	

The max size of Device Class Table is 4*199=796 (short).

TurboIMAGE Dataset File Internal Structure:

TurboIMAGE Dataset is stored as a standard file in privilege mode (-401 for standard dataset and –408 and -409 jumbo dataset). All type of dataset are stored in the following format.

USER Label 0:

Short(2B)	Content
0-1	Master: Capacity

	Detail: High-water Mark
2-3	Number of unused records
4-5	Master: unused Detail: delete chain head (**Dirty Free List**)

Record 0-N: (N is the set capacity, except for Jumbo set)

Record	Content
1	BLOCK = BIT_MAP + (MEDIA_RECORD) * BLOCKING_FACTOR BIT_MAP = (BLOCKING_FACTOR)/16Bit and round-up Master: MEDIA_RECORD = Synonym Chain Head + Path 1 Detail Chain Head + Path 2 Detail Chain Head + Path n Detail Chain Head + Data Entry (visible to you) Synonym Chain Head = Synonym Chain Count (2B) + Last Entry RecNum (4B) + First Entry RecNum (4B) Path Chain Head = Detail Chain Count (4B) + Last Entry RecNum (4B) + First Entry RecNum (4B) Detail: MEDIAL_RECORD = Path 1 Chain Link + Path 2 Chain Link + Path n Chain Link + Data Entry (visible to you) Path Chain Link = Backward RecNum (4B) + Forward RecNum (4B)
. . .	
N	Last record (may or may not be used)

OpenTURBO TurboIMAGE Internal (TI) Structure File:

This file is used by emulator as well as utilities, it is identified in CONFIG file or via program run-time option.

OpenTURBO TI file is created by TILOAD program, it contains internal structure of your original TurboIMAGE database and some OpenTURBO performance related mapping information. You can re-generate your original TurboIMAGE database schema via TILOAD –map option and verify the version of the TI file via TILOAD –v option.

In OpenTURBO environment, the TI file is part of your relational database object, it must in-sync with OpenTURBO and your relational database. You may have different versions of OpenTURBO co-exist, **but TI and OpenTURBO must be in the same version as a pair.**

TI can be existed on separate server other than the database server, it is designed for easy integration of OpenTURBO in a distributed computing environment, for scalability and for performance. So, **TI is independent to any physical database**.

In the TI file, TI structure is the root which contains base TIDB structure, set TISET structure, and item TIITEM structure.

In the TIDB data structure, DB[] contains TurboIMAGE database level control information along with some runtime control context; the TurboIMAGEBaseID and OpenTURBOBaseID are used for DUAL-MODE option, whereas CurrentSETIdx, CurrentSETNo, and CurrentSETName are strictly used for speedy set level context look-ahead and data pre-fetch.

In the TISET data structure, tiset[] is a list of sets, each set's attributes is described via TISET, the total number of sets is stored in tiset_cnt.

The set_no is in its absolute format, and the usage should be re-refresh in real-time mode via 'SELECT COUNT(*) FROM TABLE_NAME;' SQL statement, currently it contains the static usage count at the time you did the database migration.

The field data structure maintains the exact order of fields in each dataset via array subscript ascending order, and item_no is stored in it absolute format. OpenTURBO internal item_idx should be used for speedy item attributes fetch from tiitem[] list.

The path data structure keeps the primary key and all other keys along with their sort keys if there is any. OpenTURBO internal set_idx and item_idx should be used for speedy set and item attributes fetch from tiset[] and tiitem[] lists.

Current_list and CurrentTEXTList track set's ITEM-LIST, which can be changed by DBGET, DBPUT, and DBUPDATE only.

CurrentITEMIdx, CurrentITEMNo, and CurrentITEMName point to the same SEACH-KEY in three formats. STATUS[] context, chain count, forward and backward pointers, should be current SEARCH-KEY based.

In the TIITEM data structure, tiitem[] is a list of items, each item's attributes is described via TIITEM, the total number of items is stored in tiitem_cnt.

The item_no is in its absolute format, for compound item, its mapped individual column name is original item name concatenated with '_001' for the first occurrence, '_002' for the second occurrence, and so on.

```
#define SCREEN_LINE         81
#define DBNAME_SIZE         84
#define PASSWORD_SIZE       84
#define MAINT_SIZE          84
#define MAX_DB_NUM          64 /* 64 Concurrent Instants */
#define MAX_MSG_LEN         72

#define TI_MAX_CHAINS       16
#define TI_MAX_FIELDS      255
#define TI_MAX_ITEMS      1023
#define TI_MAX_PATHS        16
#define TI_MAX_SETS        199
#define TI_MAX_USERS        63
#define TI_SETN_MSIZE       16
#define TI_ITEMN_MSIZE      16
#define TI_LIST_MSIZE     4500 /* 16 Bytes(ItemNameLen) * 255(MaxField) */

typedef struct tiitem_104 { /* Items in DATASET */
        SWORD        item_count              ;
        SWORD        item_no [TI_MAX_FIELDS] ; /* ABSOLUTE() */
        SDWORD       item_idx[TI_MAX_FIELDS] ;
} TIITEM_104;

typedef struct tipath_type  { /* Set Path Details */
        SWORD        set_no         ; /* Master Data Set */
        SWORD        search_item_no ;
        SWORD        sort_item_no   ;
        SWORD        path_number    ; /* Not in the TI 301 Structure */
        SDWORD       set_idx        ;
        SDWORD       search_item_idx;
        SDWORD       sort_item_idx  ;
} TIPATH_TYPE;

typedef struct tipath_301 { /* Paths in A Detail Data Set */
        SWORD        path_count        ;
        TIPATH_TYPE  path[TI_MAX_PATHS];
} TIPATH_301;

typedef struct tipath_302 { /* Master/Detail Set KEY/SEARCH Item */
        SWORD        item_no ; /* (M)KEY or (D)SEARCH */
        SWORD        set_no  ; /* (D)Master Data Set Number */
        SDWORD       item_idx;
        SDWORD       set_idx ;
} TIPATH_302;

typedef struct tiitem {
        SWORD        item_no          ;
        SWORD        sub_length       ; /* depends upon item_type */
        SWORD        sub_count        ; /* compound item count    */
        SCHAR        item_name  [16+2];
        SCHAR        item_type  [ 2+2];
        UCHAR        read       [   8];
        UCHAR        write      [   8];
        SCHAR        column_name[ 256];
        SDWORD       byte_length      ; /* item length in bytes   */
} TIITEM;

typedef struct tipath  {
        TIPATH_301   path                      ;
        TIPATH_302   primary                   ;
} TIPATH;

typedef struct tiset {
        SDWORD       usage             ;
        SDWORD       capacity          ;
        SWORD        set_no            ;
        SWORD        entry_length      ; /* RecLen in HW         */
        SWORD        blocking_factor   ; /* # of MediaRec/Block  */
        SWORD        block_size        ; /* MPE FileRecSize (HW) */
        SWORD        full              ; /* 100% is stored 10000 */
        TIITEM_104   fields            ;
        TIPATH       paths             ;
        SCHAR        set_name    [16+2];
        SCHAR        set_type    [ 2+2];
        UCHAR        read        [   8];
        UCHAR        write       [   8];
        SWORD        block_length      ;
        SWORD        media_length      ;
```

```
        SCHAR        table_name  [ 256];
        SWORD        current_list[TI_MAX_FIELDS+1]; /* 0: # of items   */
                                                    /* 1-255: item_idx */
        UCHAR        CurrentTEXTList[TI_LIST_MSIZE+1];
        SDWORD       CurrentITEMIdx          ;
        SWORD        CurrentITEMNo           ;
        UCHAR        CurrentITEMName[TI_ITEMN_MSIZE+1];
} TISET;

typedef struct tidb {
        SWORD        mode                    ; /* Compatible Modes bit-map */
                                               /* Use First Byte Only      */
        SWORD        status                  ;
        SWORD        DBMS                    ;
        SCHAR        dbname   [  DBNAME_SIZE]; /* First 2 are BLANKS */
        SCHAR        password [PASSWORD_SIZE]; /* User Provide  */
        SCHAR        maint    [   MAINT_SIZE];
        SCHAR        pwlist   [          512]; /* From Database */
        SCHAR        option   [            8];
        SWORD        TurboIMAGEBaseID        ;
        SWORD        OpenTURBOBaseID         ;
        SDWORD       CurrentSETIdx           ;
        SWORD        CurrentSETNo            ;
        UCHAR        CurrentSETName[TI_SETN_MSIZE+1];
} TIDB;

typedef struct ti {
        TIITEM      *tiitem      [   MAX_DB_NUM];
        UDWORD       tiitem_cnt  [   MAX_DB_NUM];
        TISET       *tiset       [   MAX_DB_NUM];
        UDWORD       tiset_cnt   [   MAX_DB_NUM];
        TIDB         DB          [   MAX_DB_NUM];
        UDWORD       tidb_cnt                  ;
        UWORD        msg_filenum               ;
        SCHAR        msg_buf     [MAX_MSG_LEN+4];
        SCHAR        banner      [  SCREEN_LINE];
        LOGON        original_logon            ;
        LOGON        current_logon             ;
        UCHAR        logon_date  [            9]; /* YYYYMMDD      */
        UCHAR        logon_time  [            9]; /* HHMMSSZZ      */
        UCHAR        session     [           15]; /* Logon Session */
        UCHAR        ldev        [            9]; /* Logon ldev    */
        UCHAR        version     [            8]; /* A.00.00       */
        LT          *ValidateLt                  ;
} TI;
```

OpenTURBO Runtime Control Buffer (RCB) Internal Structure:

OpenTURBO RCB manages and controls OpenTURBO database contexts which are databases, datasets, cursors, locks, transactions, and access methods. This is for reference only and is part of iMaxsoft internal confidential document.

The otdb is one for each DBOPEN, it is corresponding to a persistent DBSVR on the server machine, and it is optional to connect to another DBSVR concurrently for handling multiple databases transaction (DBXBEGIN, DBXEND, and DBXUNDO), which is trigged via DBCONTORL Mode 7.

Each otdb has its entire otset contexts in memory for speedy context switch among datasets.

```
#define TI_BASE_MSIZE      24 /* "  INVENT.DATGROUP.IMAXSOFT"    */
                              /* should not count the leading 2  */
                              /* BLANKS                          */
#define TI_SET_MSIZE       16 /* if less than 16, must terminate */
                              /* by ; or BLANK (same for BASE)   */
#define TI_ITEM_MSIZE      16
#define TI_QUAL_MSIZE    2100 /* IN HALFWORDS - 4094B(MAX)       */
#define TI_MITEMLEN      4096 /* IN BYTES                        */
#define TI_REC_MSIZE     4756 /* IN BYTES                        */
#define TI_MSG_MSIZE      256 /* 256 HALFWORDS = 512 BYTES       */
#define TI_ERRTAB_SIZE    256
#define TI_ERRMSG_SIZE     73

static TI     OTTI;           /* 64 DBOPENs per client processes */
static SDWORD otticnt=0;      /* 64 DBOPENs per client processes */
static SDWORD CurrentDBIdx=0;
static SDWORD CurrentCALL;
static SDWORD CurrentDBXBEGIN=FALSE;
static UCHAR  ERRMsg[TI_ERRTAB_SIZE*TI_ERRMSG_SIZE];
static SDWORD ERRNum[TI_ERRTAB_SIZE];
static SCHAR  csridNULL[DB_CURSOR_ID_LEN];
static SDWORD ChainGetModeChange; /* TRUE or FALSE */
static SDWORD DBGETBufLength=0; /* DUAL MODE Only 2002-06-11 */

typedef struct otdb_type {
       SCHAR      dbname     [DBNAME_SIZE]; /* DBNAME_SIZE = 84         */
                                            /* True Name -> db.grp.acct */
                                            /* No ' ', ';', leading ' ' */
       SCHAR      owner      [DBNAME_SIZE];
       SCHAR      tifile     [DBNAME_SIZE];
       SCHAR      rwfile     [DBNAME_SIZE]; /* reserved word fname */
       SCHAR      erfile     [DBNAME_SIZE]; /* error message fname */
       SCHAR      host       [DBNAME_SIZE];
       SCHAR      service    [DBNAME_SIZE];
       SDWORD     osRDBMS                 ; /* 513 for HPUX+ORACLE */
                                            /* 772 for 2000+SQLSVR */
       SCHAR      osLogon    [DBNAME_SIZE];
       SCHAR      rdbLogon   [DBNAME_SIZE];
       SCHAR      svrPgm     [DBNAME_SIZE];
       SCHAR      hDB        [DB_HANDLE_LEN];
       SCHAR      hDBLOCK    [DB_HANDLE_LEN];
       SWORD      DBLocked                ; /* Database Level LOCK */
       SDWORD     LOCKTryCnt              ; /* Max Retry UnconLOCK */
       SDWORD     LOCKPause               ; /* Retry interval (ss) */
       SDWORD     OT_CIUPDATE             ; /* TRUE or FALSE       */
       SDWORD     OT_DUALMODE             ; /* TRUE or FALSE       */
       SDWORD     ALLOW_DBXBEGIN          ; /* TRUE or FALSE       */
       SDWORD     DBXBEGIN                ; /* TRUE or FALSE       */
       SDWORD     SDKCallCnt              ;
} OTDB;

#define SQLSTMT_LEN   12288
#define SQLLIST_LEN    8192
#define SQLWHERE_LEN   4096
#define SQLVALUE_LEN   4096
#define SDKBUF_LEN    30000 /* contains all SEQ_NO for a specific path */
                            /* The SDK buffer max size is 30000 bytes  */
#define SDKROW_LEN     8192
#define DB_ROWID_LEN     27 /* For ORACLE - 18 */
```

```
#define OT_NO             0
#define OT_YES            1
#define OT_STAY           2

/*********************************************************************/
/* lockMode: if != 0 then there is a LOCK in effect for this dataset  */
/* lockDesc:                                                          */
/*     [ 0]: # of descriptor for the dataset                          */
/*     [ 1]: length (HW) for the descriptor                           */
/*     [ 8]: 16 Bytes - SET_NO (set_idx)                              */
/*     [ 8]: 16 Bytes - ITEM_NO (item_idx)                            */
/*     [ 2]: TI_LOCK_EQOP, TI_LOCK_LEOP, TO_LOCK_GEOP                 */
/*     [ n]: n*2 Bytes for the ITEM_VALUE                             */
/*                                                                    */
/*     If [1] == 9 then it is a dataset level lock                    */
/*********************************************************************/
typedef struct otset_type {
       SCHAR      SQLList    [     SQLLIST_LEN]; /* 8192 */
       SCHAR      SQLWhere   [    SQLWHERE_LEN]; /* 1024 */
       SDWORD     SQLWhereLen                   ;
       SDWORD     GETMode                       ; /* Current DBGET Mode */
       SDWORD     ncols                         ;
       SDWORD     nrows                         ;
       SDWORD     more                          ;
       SDWORD     current                       ;
       SDWORD     BULKncols                     ;
       SDWORD     BULKnrows                     ;
       SDWORD     BULKmore                      ;
       SDWORD     BULKcurrent                   ;
       SCHAR     *rowptr                        ;
       SCHAR     *Browptr                       ;
       SCHAR     *Frowptr                       ;
       SCHAR     *FIRSTptr                      ;
       SCHAR     *LASTptr                       ;
       SCHAR      csrid      [DB_CURSOR_ID_LEN];
       SDWORD     IMAXSOFT13_SEQ_NO             ; /* Current SeqNO/RecNUM */
       SDWORD     FROM_BEGIN_FILE               ; /* For MODE 2/3 Only    */
       SCHAR     *buf                           ;
       SCHAR     *BULKbuf                       ;
       SCHAR     *BULKptr                       ;
       SDWORD     CURSOR_BUFCNT                 ;
       SDWORD     CHRONOLOGICAL_ON              ; /* TRUE or FALSE */
       SDWORD     BULKCHAINGET_ON               ; /* TRUE or FALSE */
       SDWORD     IGNORE_CHAINSTATUS_ON         ; /* TRUE or FALSE */
       SWORD      lockDesc[TI_QUAL_MSIZE+1]     ;
       SWORD      lockMode                      ;
} OTSET;

typedef struct ot_type {
       OTDB       otdb [MAX_DB_NUM];
       OTSET     *otset[MAX_DB_NUM];
} OT_TYPE;

static OT_TYPE OTOT;

/*****************************************************/
/* Global Variable Declaration for RESERVE WORD (RDBM) */
/*****************************************************/
#define RESERVE_SUFFIX_MSIZE 20

static SDWORD  RESERVE_CNT;
static SCHAR   RESERVE_SUFFIX[RESERVE_SUFFIX_MSIZE+1];
static SCHAR   RESERVE_WORD[1024][256];
```

2. OpenTURBO QUICK START - simple steps to jump-start your HP3000 migration.

   STEP 1: On HP e3000, the ALL script file:

   1. HP3000 – Generates the corresponding ORACLE schema from TurboIMAGE database
   2. HP3000 – Generates the OpenTURBO root-file which contains all cross-database and cross-platform mapping rules
   3. HP3000 – Unloads data from TurboIMAGE into standard flat files and generates ORACLE table loading script one per dataset
   4. HP3000 – Generates ftp script to transfer 1-3 generated files to the target open platform HP9000 HP-UX

```
comment ----------------------------------------------------
comment Turn ON(=1)/OFF(=0) Client Debugging Facility
comment
echo
echo ==> Debugging Facility Setting ....
setvar ltdbg1 0
setvar ltdbg2 0
setvar ltdbg3 0
setvar ltdbg4 0
setvar ltdbgout "$stdlist"
comment ----------------------------------------------------
comment Create TurboIMAGE and OpenTURBO database structure
comment and TurboIMAGE/ORACLE mapping rules binary file
comment
echo
echo ==> OpenTURBO TI Rootfile Creation ....
purge MUSICti
genti
comment ----------------------------------------------------
comment Generate ORACLE schema for creating tables,
comment indexes, and OpenTURBO mapping information
comment
echo
echo ==> Generating ORACLE schema without Automatic dataset ....
schema1
echo
echo ==> Generating ORACLE schema with Automatic dataset ....
schema2
comment ----------------------------------------------------
comment Unload all datasets from TurboIMAGE into flat
comment files (DMDAT group) and generate all ORACLE
comment table load script files (DMCLT group)
comment
echo
echo ==> Dataset Unloading ....
allsets
comment ----------------------------------------------------
comment Generate ftp job file for transferring all files
comment from HP3000 to HP9000
comment
echo
echo ==> Generating ftp Job File ....
genftp
reset music.tidata.imaxsoft
echo
echo ==================================
echo Process is completed successfully.
echo ==================================
echo
```

```
Script Files:

   File genti:

      tiload.bin;info='-dmusic.tidata.imaxsoft -tMUSICti
      -rreserve/oracle'

   File schema1 – without Automatic datasets:

      purge smusic1.otschema
      otdrv.bin;info='-dmusic.tidata.imaxsoft -t1 -rreserve.oracle
      -c -recnum -osmusic1.otschema'

   File schema2 – with Automatic datasets:

      purge smusic2.schema
      otdrv.bin;info='-dmusic.tidata.imaxsoft -t2 -rreserve.oracle
      -c -recnum -osmusic2.otschema'

   File allsets – unload all datasets in BINARY:

      setvar ltdbg2 0
      setvar ltdbgout '$stdlist'
      otdrv.bin;info= ' &
      -d  music.tidata.imaxsoft &
      -ti MUSICti &
      -v  reserve.oracle &
      -t  20 &
      -r  OTDATA &
      -l  OTSCRIPT &
      -o  otcopy &
      -w  otwork &
      -c &
      -recnum &
      -s  @'

   File genftp:

      genftp.bin;info= ' &
      -d MUSIC &
      -r OTDATA &
      -l OTSCRIPT &
      -k OTSCHEMA &
      -o JLDFTP &
      -h 207.92.64.8 &
      -p musicdemo/data &
      -u lee/gzz0122
```

STEP 2: On HP9000 HP-UX, the ALL script file:

1. HP9000 – Creates ORACLE table-space analysis report for disk spaces allocation
2. HP9000 – Creates corresponding ORACLE tables, indexes, triggers, sequences, and constrains
3. HP9000 – Replicates TurboIMAGE access security to the ORACLE database
4. HP9000 – Loads data into ORACLE tables

```
echo
echo ------------------------------------------------
echo - OpenTURBO ORACLE Script Pre-process Begins .. -
```

```
echo -------------------------------------------------
echo
echo --Processing ORACLE schema script for sqlplus ....
createOTBuildDBScript
echo
echo --Buliding ORACLE database ....
OTBuildDB.EXECUTE
echo
echo --Re-fining ORACLE load script for speedy loading ....
createOTLoadDBScript
echo
echo --Replicating TurboIMAGE security into ORACLE database ....
createSECScript
echo
echo --Moving ORACLE script files into working directory ....
movePROD
echo
echo -------------------------------------------------
echo - OpenTURBO ORACLE Database Creation Begins ... -
echo -------------------------------------------------
echo
echo --Switching to working directory
cd ../data
echo
echo --Loading ORACLE database ....
OTLoadDB.EXECUTE
echo
echo --Creating ORACLE database security ....
OTCreateSEC.EXECUTE
echo
echo -------------------------------------------------
echo - OpenTURBO Post-creation Validation reports .. -
echo -------------------------------------------------
otDBReport
echo
echo -------------------------------------------------
echo - OpenTURBO Post-creation Checksum Settings ... -
echo -------------------------------------------------
rSetCheckSum
echo
echo -------------------------------------------------
echo - OpenTURBO ORACLE Database Creation completed. -
echo -------------------------------------------------
```

Script Files:

File createOTBuildDBScript:

```
/users/lee/bin/otgenDB -uOT/JO \
                       -dMUSIC.TIDATA.IMAXSOFT \
                       -k../data/SMUSIC2.OTSCHEMA \
                       -pHPNDX \
                       -sMUSIC
```

File createOTLoadDBScript:

```
/users/lee/bin/otgenLOAD -u ot/JO                  \
                         -d MUSIC.TIDATA.IMAXSOFT \
                         -t @                      \
                         -c NOCACHE
```

File createSECScript: you must put RESERVE.ORACLE file in directory $_OTB_ROOT/conf or use -r to identify the location of file RESERVE.ORACLE.

```
/users/lee/bin/tigenSEC -d MUSIC.TIDATA.IMAXSOFT \
                        -t MUSIC.TIDATA.IMAXSOFT \
                        -r /users/lee/conf/RESERVE.ORACLE  \
                        -p MUSIC \
                        -s MUSIC
```

File movePROD:

```
cp OTBuildDB.EXECUTE    /users/lee/musicdemo/data/.
cp OTLoadDB.DROP        /users/lee/musicdemo/data/.
cp OTLoadDB.GEN         /users/lee/musicdemo/data/.
cp OTLoadDB.SQLLDR      /users/lee/musicdemo/data/.
cp OTLoadDB.EXECUTE     /users/lee/musicdemo/data/.
cp OTCreateSEC.EXECUTE  /users/lee/musicdemo/data/.
cp OTCreateSEC.CLASS    /users/lee/musicdemo/data/.
cp otDBReport           /users/lee/musicdemo/data/.
cp rSetCheckSum         /users/lee/musicdemo/data/.
```

Other script files are automatically generated by OpenTURBO, run them as they are and do not edit them.

STEP 3: On HP9000, starts OpenTURBO listener process, you must have super user capability and login to the environment that allows you to access the target ORALCE sid and with proper path setup for OpenTURBO server programs and libraries.

Start the Listener: $_OTB_BIN/listner 32601

Stop the Listener: $_OTB_BIN/lanutil 32601

```
LANUTIL (A.06.00.00) LeeTech Software Inc. Copyright 1993-, All Rights
Reserved.

HOST:[127.0.0.1]  APPLICATION:[32601]

Commands: LIST    - shows all connected users.
          KILL id - kills the specified user.
          STOPALL - terminates listener and all users.
          HOST id - sets to new host node name.
          APPL id - sets to new application name.
          SETQ qname #servers
                  - sets # of standby servers for a queue
          EXIT    - ends the LANUTIL.
LANUTIL>> stopall
The listener and all users have been shutdown successfully.
```

SETP 4: On HP3000, accesses ORACLE via QUERY.PUB.SYS

1. Standard way to run QUERY.PUB.SYS, first make a copy of QUERY.PUB.SYS into QUERY.BIN.IMS, create a command file as following and execute it. The IMS can be any account, and the BIN can be any group. You must also have a CONFIG file setup for database definitions.

   Sample CONFIG File:

```
// This is used by QUERY.PUB.SYS - 2002-6-25 Lee Tsai
//
// For:  SYSTEM LEVEL CONFIGURATION FILE for OpenTURBO
//       This is used by HP/3000 Client SDK Version
// Date: 2002/04/18
//
// 515 is for HPUX and ORACLE
//
OT_TI_DBNAME = MUSIC.TIDATA.IMAXSOFT
```

```
MUSIC.TIDATA.IMAXSOFT {
MUSIC.TIDATA {
MUSIC {
   OT_IMAGEMODE         = OFF
   OT_ROOT_FILE         = MUSICti.otcmd.imaxsoft
   OT_RESERVE_WORD_FILE = RESERVE.ORACLE.imaxsoft
   OT_ERROR_FILE        = OTERROR.ORACLE.imaxsoft
   OT_HOST              = 207.92.64.8
   OT_SERVICE           = 32601
   OT_OS_RDBMS          = 515
   OT_RDB_LOGON         = music.tidata.imaxsoft/hpndx
   OT_SDK_SERVER_PGM    = /users/lee/lbin/dbsvr
   OT_CIUPDATE          = ON
   OT_DUALMODE          = OFF
   TI_DUALMODE_HOST     = 207.92.64.66
   TI_DUALMODE_SERVICE  = 32608
   TI_DUALMODE_PRM      = DMDRV.BIN.IMAXSOFT
   OT_LOCKWAIT_CYCLE    = 60
   OT_TRX_THRESHOLD     = 10
OT_DETAILSETNAME = @ [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
//
// For Detail Dataset ITMDTL Only
//
OT_DETAILSETNAME = ITMDTL [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
}
//
// Second Database: In the same ORACLE instance
//
OT_TI_DBNAME = INVENT.DATA.IMS
INVENT.DATA.IMS {
INVENT.DATA {
INVENT {
   OT_IMAGEMODE         = OFF
   OT_ROOT_FILE         = ti2.ti.ims
   OT_RESERVE_WORD_FILE = RESERVE.ORACLE.ims
   OT_ERROR_FILE        = OTERROR.ORACLE.ims
   OT_HOST              = 207.92.64.8
   OT_SERVICE           = 32601
   OT_OS_RDBMS          = 515
   OT_RDB_LOGON         = invent_data_ims/diqzio
   OT_SDK_SERVER_PGM    = /users/lee/lbin/dbsvr
   OT_CIUPDATE          = ON
   OT_DUALMODE          = OFF
OT_DETAILSETNAME = @ [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
}
//
// Third Database : in the same ORACLE instance
//
OT_TI_DBNAME = CONSDB.DATABASE.LITTLEOC
CONSDB.DATABASE.LITTLEOC {
CONSDB.DATABASE {
CONSDB {
   OT_IMAGEMODE         = OFF
   OT_ROOT_FILE         = ti3.ti.ims
   OT_RESERVE_WORD_FILE = RESERVE.ORACLE.ims
   OT_ERROR_FILE        = OTERROR.ORACLE.ims
   OT_HOST              = 207.92.64.8
   OT_SERVICE           = 32601
   OT_OS_RDBMS          = 515
   OT_RDB_LOGON         = consdb_database_littleoc/xjinyw
   OT_SDK_SERVER_PGM    = /users/lee/lbin/dbsvr
   OT_CIUPDATE          = ON
   OT_DUALMODE          = OFF
```

```
OT_DETAILSETNAME = @ [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
OT_DETAILSETNAME = MBRHISTDETL [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
OT_DETAILSETNAME = LOCINFODETL [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
}
//
// Fourth Database: is a SQL SERVER database
// 772 is for SQL_SERVER 2000 on Windows 2000
//
OT_TI_DBNAME = INVENT_DATA_MOULTON
INVENT.DATA.MOULTON {
INVENT.DATA {
INVENT {
   OT_ROOT_FILE         = ti4.ti.ims
   OT_RESERVE_WORD_FILE = RESERVE.SQLSVR.ims
   OT_ERROR_FILE        = OTERROR.SQLSVR.ims
   OT_HOST              = 207.92.64.6
   OT_SERVICE           = 32601
   OT_OS_RDBMS          = 772
   OT_RDB_LOGON         = INVENT_DATA_MOULTON/sa.lee0122
   OT_SDK_SERVER_PGM    = e:\leetech\dbsvr.exe
   OT_CIUPDATE          = ON
   OT_DUALMODE          = OFF
OT_DETAILSETNAME = @ [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
}
```

OTQ Command File using OpenTURBO Standard Emulator Library, OTQRY:

```
file config=configq.lee.ims
run query.bin.ims;xl='otqry.a.imaxsoft,tidrv.a.imaxsoft,ltxl.pub.imaxsoft'
```

2. Run QUERY.PUB.SYS in OpenTURBO debugging mode.

   OTQDBG Command File using OpenTURBO Debugger Emulator Library, OTQRYDBG:

```
setvar ltdbg1 1
setvar ltdbg2 1
setvar ltdbg3 1
setvar ltdbg4 1
setvar ltdbg5 1
setvar ltdbg6 1
setvar ltdbg8 1
setvar ltdbg17 0
setvar ltdbg18 0
setvar ltdbg19 0
setvar ltdbgout '$stdlist'
file config=configq.lee.ims
run query.bin.ims;xl='otqrydbg.a.imaxsoft,tidrv.a.imaxsoft,ltxl.pub.imaxsoft'
```

STEP 5: On both HP3000 and HP9000, accesses ORACLE via TIDRV, the OpenTURBO Test Driver program.

HP3000 TIDRV Run Script File: you must put RESERVE.ORACLE file in your login account, or use file equation command :FILE RESERVE.ORACLE = to locate the file, or use -r to identify the location of file RESERVE.ORACLE.

```
parm xl='NOOT'
setvar xlflag '!xl'
setvar ltdbg1 0
setvar ltdbg2 0
setvar ltdbg3 0
setvar ltdbg4 1
setvar ltdbgout '$stdlist'
purge outupd.output
if (xlflag = 'OT') then
   file config=config.lee.ims
   run tidrv.bin;xl='otxl.a,tidrv.a,ltxl.pub.imaxsoft'; &
   info='-iinupd.lee -ooutupd.output'
else
   run tidrv.bin;xl='tidrv.a,ltxl.pub.imaxsoft'; &
   info='-iinupd.lee -ooutupd.output'
endif
```

HP9000 TIDRV Run Script File: you must put RESERVE.ORACLE file in directory $_OTB_ROOT/conf or use -r to identify the location of file RESERVE.ORACLE.

```
export LTDBGOUT=-
$_OTB_BIN/tidrv -iinupd –ooutupd
```

TIDRV Command Input File (HP3000 and HP9000): refer to above for identifying RESERVE.ORACLE file.

```
LOADTI,ti1.ti
// ** DBPUT to Detail Dataset - PURCHASE
DBOPEN,INVENT.DATA;,;,3
DEBUGOUT /users/lee/tmp/lee.dbg
DEBUG19 ON
DEBUG18 ON
DEBUG17 ON
// ** Backward Serial DBGET 4 Records from Detail Dataset PURCHASE
// ** and DBDELETE all 4 Records
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBDELETE,0,PURCHASE;,1
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBDELETE,0,PURCHASE;,1
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBDELETE,0,PURCHASE;,1
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBDELETE,0,PURCHASE;,1
//
// ** DBPUT using @ Item LIST (Record 1)
DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_1,POR_KEY_1,1,1,1,1,1,1,&
REV_LOT,REQ_NO,COMMENT_IMS,1,USER_IMS,20020526,105000'
//
// ** DBPUT using Partial Item LIST (Record 2)
DBPUT,0,PURCHASE;,1,LIST='USER,CC-PNLC-PO,DATE-UPDT,por-key;',&
DATA='USER_IMS,CC_PNLC_PO_2,20020526,POR_KEY_2'
//
// ** DBPUT using @ Item LIST (Record 3)
DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_3,POR_KEY_3,1,1,1,1,1,1,&
REV_LOT,REQ_NO,COMMENT_IMS,1,USER_IMS,20020526,105000'
//
// ** DBPUT using Partial Item LIST (Record 4)
DBPUT,0,PURCHASE;,1,LIST='USER,CC-PNLC-PO,DATE-UPDT,por-key;',&
DATA='USER_IMS,CC_PNLC_PO_4,20020526,POR_KEY_4'
//
// Rewind Dataset PURCHASE
// ** Backward Serial DBGET 4 Records from Detail Dataset PURCHASE
DBCLOSE,0,PURCHASE,2
REPEAT 4
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
```

```
//
// Rewind Dataset PURCHASE
// ** DBUPDATE those 4 just added records
DBCLOSE,0,PURCHASE,2
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBUPDATE,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO,POR_KEY,-9,-9,-9,-9,-9,-9,&
REV_LOT,REQ_NO,IMAXSOFT,-9,IMAXSOFT,20020526,141414'
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBUPDATE,0,PURCHASE;,1,LIST='USER,qty-ord,DATE-UPDT,qty-recd;',&
DATA='IMAXSOFT,-444,20020525,-666'
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBUPDATE,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO,POR_KEY,+9,+9,+9,+9,+9,+9,&
REV_LOT,REQ_NO,IMAXSOFT,+9,IMAXSOFT,20020526,141414'
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBUPDATE,0,PURCHASE;,1,LIST='USER,qty-ord,DATE-UPDT,qty-recd;',&
DATA='IMAXSOFT,888,20020525,666'
//
// Rewind Dataset PURCHASE
// ** Backward Serial DBGET 4 Records from Detail Dataset PURCHASE
DBCLOSE,0,PURCHASE,2
REPEAT 4
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBCLOSE,0,,1
UNLOADTI
//
```

TIDRV Result Output File (HP3000 and HP9000):

```
=>> Repeat[1] TICommand[LOADTI,ti1.ti]
=>> Repeat[1] TICommand[DBOPEN,INVENT.DATA;,;,3]
DBOPEN,INVENT.DATA;,;,3,status[1]=0,status[2]=64
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>MOXIMG04006BRGL   N 2001
POR-KEY             1X20  =>MOXI2001
VEND-NO             1I2   =>+0000001001
QTY-ORD             1I2   =>+0000000100
DATE-ORD            1I2   =>+0000981123
DATE-PROM           1I2   =>+0019981130
QTY-RECD            1I2   =>+0000000002
DATE-LAST           1I2   =>+0019981123
REV-LOT             1X10  =>A111
REQ-NO              1X16  =>
COMMENT             1X30  =>
UNIT-COST           1I4   =>0000000000000135000
USER                1X8   =>
DATE-UPDT           1I2   =>+0000000000
TIME-UPDT           1I2   =>+0000000000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[114]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBDELETE,0,PURCHASE;,1]
DBDELETE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
status[1]   =[0]
status[2]   =[0]
status[3-4] =[114]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>BLUEBXIU2036      N 154148
```

```
POR-KEY            1X20  =>BLUE154148
VEND-NO            1I2   =>+0000001009
QTY-ORD            1I2   =>+0000027000
DATE-ORD           1I2   =>+0019980715
DATE-PROM          1I2   =>+0019980826
QTY-RECD           1I2   =>+0000000000
DATE-LAST          1I2   =>+0000000000
REV-LOT            1X10  =>
REQ-NO             1X16  =>
COMMENT            1X30  =>8/98
UNIT-COST          1I4   =>-000000000000000012
USER               1X8   =>NANCY
DATE-UPDT          1I2   =>+0000981006
TIME-UPDT          1I2   =>+0011140250
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[113]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBDELETE,0,PURCHASE;,1]
DBDELETE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
status[1]   =[0]
status[2]   =[0]
status[3-4] =[113]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>BLUEBXIU2013      N 154033
POR-KEY            1X20  =>BLUE154033
VEND-NO            1I2   =>+0000001001
QTY-ORD            1I2   =>+0000020000
DATE-ORD           1I2   =>+0019980805
DATE-PROM          1I2   =>+0019980828
QTY-RECD           1I2   =>+0000000000
DATE-LAST          1I2   =>+0000000000
REV-LOT            1X10  =>
REQ-NO             1X16  =>
COMMENT            1X30  =>11,400, 1,400 overs
UNIT-COST          1I4   =>0000000000000000000
USER               1X8   =>NANCY
DATE-UPDT          1I2   =>+0000981006
TIME-UPDT          1I2   =>+0011140250
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[112]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBDELETE,0,PURCHASE;,1]
DBDELETE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
status[1]   =[0]
status[2]   =[0]
status[3-4] =[112]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>BLUEBXIU2013      N 152239
POR-KEY            1X20  =>BLUE152239
```

```
VEND-NO            1I2   =>+0000001001
QTY-ORD            1I2   =>+0000010000
DATE-ORD           1I2   =>+0019980930
DATE-PROM          1I2   =>+0019980930
QTY-RECD           1I2   =>+0000000000
DATE-LAST          1I2   =>+0000000000
REV-LOT            1X10  =>
REQ-NO             1X16  =>
COMMENT            1X30  =>Rush
UNIT-COST          1I4   =>0000000000000000000
USER               1X8   =>NANCY
DATE-UPDT          1I2   =>+0000981006
TIME-UPDT          1I2   =>+0011140250
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[111]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBDELETE,0,PURCHASE;,1]
DBDELETE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
status[1]   =[0]
status[2]   =[0]
status[3-4] =[111]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1]
TICommand[DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_1,POR_KEY_1,1,1,1,1,1,1,R
EV_LOT,REQ_NO,COMMENT_IMS,1,USER_IMS,20020526,105000']
DBPUT: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO_1
POR-KEY            1X20  =>POR_KEY_1
VEND-NO            1I2   =>+0000000001
QTY-ORD            1I2   =>+0000000001
DATE-ORD           1I2   =>+0000000001
DATE-PROM          1I2   =>+0000000001
QTY-RECD           1I2   =>+0000000001
DATE-LAST          1I2   =>+0000000001
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
COMMENT            1X30  =>COMMENT_IMS
UNIT-COST          1I4   =>0000000000000000001
USER               1X8   =>USER_IMS
DATE-UPDT          1I2   =>+0020020526
TIME-UPDT          1I2   =>+0000105000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[555]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBPUT,0,PURCHASE;,1,LIST='USER,CC-PNLC-PO,DATE-UPDT,por-
key;',DATA='USER_IMS,CC_PNLC_PO_2,20020526,POR_KEY_2']
DBPUT: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=USER,CC-PNLC-PO,DATE-UPDT,por-key;
DATA BUFFER Begin ====================
USER               1X8   =>USER_IMS
CC-PNLC-PO         1X36  =>CC_PNLC_PO_2
DATE-UPDT          1I2   =>+0020020526
POR-KEY            1X20  =>POR_KEY_2
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[34]
status[3-4] =[557]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
```

```
=>> Repeat[1]
TICommand[DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_3,POR_KEY_3,1,1,1,1,1,1,R
EV_LOT,REQ_NO,COMMENT_IMS,1,USER_IMS,20020526,105000']
DBPUT: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_3
POR-KEY             1X20  =>POR_KEY_3
VEND-NO             1I2   =>+0000000001
QTY-ORD             1I2   =>+0000000001
DATE-ORD            1I2   =>+0000000001
DATE-PROM           1I2   =>+0000000001
QTY-RECD            1I2   =>+0000000001
DATE-LAST           1I2   =>+0000000001
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>COMMENT_IMS
UNIT-COST           1I4   =>0000000000000000001
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000105000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[559]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBPUT,0,PURCHASE;,1,LIST='USER,CC-PNLC-PO,DATE-UPDT,por-
key;',DATA='USER_IMS,CC_PNLC_PO_4,20020526,POR_KEY_4']
DBPUT: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=USER,CC-PNLC-PO,DATE-UPDT,por-key;
DATA BUFFER Begin ====================
USER                1X8   =>USER_IMS
CC-PNLC-PO          1X36  =>CC_PNLC_PO_4
DATE-UPDT           1I2   =>+0020020526
POR-KEY             1X20  =>POR_KEY_4
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[34]
status[3-4] =[561]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBCLOSE,0,PURCHASE,2]
DBCLOSE,INVENT.DATA;[0],PURCHASE[15],2,db_status[1]=0
=>> Repeat[4] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_4
POR-KEY             1X20  =>POR_KEY_4
VEND-NO             1I2   =>+0000000000
QTY-ORD             1I2   =>+0000000000
DATE-ORD            1I2   =>+0000000000
DATE-PROM           1I2   =>+0000000000
QTY-RECD            1I2   =>+0000000000
DATE-LAST           1I2   =>+0000000000
REV-LOT             1X10  =>
REQ-NO              1X16  =>
COMMENT             1X30  =>
UNIT-COST           1I4   =>0000000000000000000
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000000000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[561]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
```

```
=>> Repeat[3] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_3
POR-KEY             1X20  =>POR_KEY_3
VEND-NO             1I2   =>+0000000001
QTY-ORD             1I2   =>+0000000001
DATE-ORD            1I2   =>+0000000001
DATE-PROM           1I2   =>+0000000001
QTY-RECD            1I2   =>+0000000001
DATE-LAST           1I2   =>+0000000001
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>COMMENT_IMS
UNIT-COST           1I4   =>0000000000000000001
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000105000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[559]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[2] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_2
POR-KEY             1X20  =>POR_KEY_2
VEND-NO             1I2   =>+0000000000
QTY-ORD             1I2   =>+0000000000
DATE-ORD            1I2   =>+0000000000
DATE-PROM           1I2   =>+0000000000
QTY-RECD            1I2   =>+0000000000
DATE-LAST           1I2   =>+0000000000
REV-LOT             1X10  =>
REQ-NO              1X16  =>
COMMENT             1X30  =>
UNIT-COST           1I4   =>0000000000000000000
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000000000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[557]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_1
POR-KEY             1X20  =>POR_KEY_1
VEND-NO             1I2   =>+0000000001
QTY-ORD             1I2   =>+0000000001
DATE-ORD            1I2   =>+0000000001
DATE-PROM           1I2   =>+0000000001
QTY-RECD            1I2   =>+0000000001
DATE-LAST           1I2   =>+0000000001
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>COMMENT_IMS
UNIT-COST           1I4   =>0000000000000000001
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000105000
DATA BUFFER End   ====================
```

```
status[1]   =[0]
status[2]   =[80]
status[3-4] =[555]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBCLOSE,0,PURCHASE,2]
DBCLOSE,INVENT.DATA;[0],PURCHASE[15],2,db_status[1]=0
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_4
POR-KEY             1X20  =>POR_KEY_4
VEND-NO             1I2   =>+0000000000
QTY-ORD             1I2   =>+0000000000
DATE-ORD            1I2   =>+0000000000
DATE-PROM           1I2   =>+0000000000
QTY-RECD            1I2   =>+0000000000
DATE-LAST           1I2   =>+0000000000
REV-LOT             1X10  =>
REQ-NO              1X16  =>
COMMENT             1X30  =>
UNIT-COST           1I4   =>0000000000000000000
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000000000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[561]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1]
TICommand[DBUPDATE,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO,POR_KEY,-9,-9,-9,-9,-
9,-9,REV_LOT,REQ_NO,IMAXSOFT,-9,IMAXSOFT,20020526,141414']
DBUPDATE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO
POR-KEY             1X20  =>POR_KEY
VEND-NO             1I2   =>-0000000009
QTY-ORD             1I2   =>-0000000009
DATE-ORD            1I2   =>-0000000009
DATE-PROM           1I2   =>-0000000009
QTY-RECD            1I2   =>-0000000009
DATE-LAST           1I2   =>-0000000009
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>IMAXSOFT
UNIT-COST           1I4   =>-000000000000000009
USER                1X8   =>IMAXSOFT
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000141414
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[561]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_3
POR-KEY             1X20  =>POR_KEY_3
VEND-NO             1I2   =>+0000000001
QTY-ORD             1I2   =>+0000000001
DATE-ORD            1I2   =>+0000000001
DATE-PROM           1I2   =>+0000000001
```

```
QTY-RECD           1I2   =>+0000000001
DATE-LAST          1I2   =>+0000000001
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
COMMENT            1X30  =>COMMENT_IMS
UNIT-COST          1I4   =>0000000000000000001
USER               1X8   =>USER_IMS
DATE-UPDT          1I2   =>+0020020526
TIME-UPDT          1I2   =>+0000105000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[559]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBUPDATE,0,PURCHASE;,1,LIST='USER,qty-ord,DATE-UPDT,qty-
recd;',DATA='IMAXSOFT,-444,20020525,-666']
DBUPDATE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=USER,qty-ord,DATE-UPDT,qty-recd;
DATA BUFFER Begin ====================
USER               1X8   =>IMAXSOFT
QTY-ORD            1I2   =>-0000000444
DATE-UPDT          1I2   =>+0020020525
QTY-RECD           1I2   =>-0000000666
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[10]
status[3-4] =[559]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO_2
POR-KEY            1X20  =>POR_KEY_2
VEND-NO            1I2   =>+0000000000
QTY-ORD            1I2   =>+0000000000
DATE-ORD           1I2   =>+0000000000
DATE-PROM          1I2   =>+0000000000
QTY-RECD           1I2   =>+0000000000
DATE-LAST          1I2   =>+0000000000
REV-LOT            1X10  =>
REQ-NO             1X16  =>
COMMENT            1X30  =>
UNIT-COST          1I4   =>0000000000000000000
USER               1X8   =>USER_IMS
DATE-UPDT          1I2   =>+0020020526
TIME-UPDT          1I2   =>+0000000000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[557]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1]
TICommand[DBUPDATE,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO,POR_KEY,+9,+9,+9,+9,+9
,+9,REV_LOT,REQ_NO,IMAXSOFT,+9,IMAXSOFT,20020526,141414']
DBUPDATE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO
POR-KEY            1X20  =>POR_KEY
VEND-NO            1I2   =>+0000000009
QTY-ORD            1I2   =>+0000000009
DATE-ORD           1I2   =>+0000000009
DATE-PROM          1I2   =>+0000000009
QTY-RECD           1I2   =>+0000000009
DATE-LAST          1I2   =>+0000000009
```

```
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
COMMENT            1X30  =>IMAXSOFT
UNIT-COST          1I4   =>0000000000000000009
USER               1X8   =>IMAXSOFT
DATE-UPDT          1I2   =>+0020020526
TIME-UPDT          1I2   =>+0000141414
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[557]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO_1
POR-KEY            1X20  =>POR_KEY_1
VEND-NO            1I2   =>+0000000001
QTY-ORD            1I2   =>+0000000001
DATE-ORD           1I2   =>+0000000001
DATE-PROM          1I2   =>+0000000001
QTY-RECD           1I2   =>+0000000001
DATE-LAST          1I2   =>+0000000001
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
COMMENT            1X30  =>COMMENT_IMS
UNIT-COST          1I4   =>0000000000000000001
USER               1X8   =>USER_IMS
DATE-UPDT          1I2   =>+0020020526
TIME-UPDT          1I2   =>+0000105000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[555]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBUPDATE,0,PURCHASE;,1,LIST='USER,qty-ord,DATE-UPDT,qty-
recd;',DATA='IMAXSOFT,888,20020525,666']
DBUPDATE: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[1]
list=USER,qty-ord,DATE-UPDT,qty-recd;
DATA BUFFER Begin ====================
USER               1X8   =>IMAXSOFT
QTY-ORD            1I2   =>+0000000888
DATE-UPDT          1I2   =>+0020020525
QTY-RECD           1I2   =>+0000000666
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[10]
status[3-4] =[555]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBCLOSE,0,PURCHASE,2]
DBCLOSE,INVENT.DATA;[0],PURCHASE[15],2,db_status[1]=0
=>> Repeat[4] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO
POR-KEY            1X20  =>POR_KEY
VEND-NO            1I2   =>-0000000009
QTY-ORD            1I2   =>-0000000009
DATE-ORD           1I2   =>-0000000009
DATE-PROM          1I2   =>-0000000009
QTY-RECD           1I2   =>-0000000009
DATE-LAST          1I2   =>-0000000009
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
```

```
COMMENT            1X30  =>IMAXSOFT
UNIT-COST          1I4   =>-000000000000000009
USER               1X8   =>IMAXSOFT
DATE-UPDT          1I2   =>+0020020526
TIME-UPDT          1I2   =>+0000141414
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[561]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[3] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO_3
POR-KEY            1X20  =>POR_KEY_3
VEND-NO            1I2   =>+0000000001
QTY-ORD            1I2   =>-0000000444
DATE-ORD           1I2   =>+0000000001
DATE-PROM          1I2   =>+0000000001
QTY-RECD           1I2   =>-0000000666
DATE-LAST          1I2   =>+0000000001
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
COMMENT            1X30  =>COMMENT_IMS
UNIT-COST          1I4   =>000000000000000001
USER               1X8   =>IMAXSOFT
DATE-UPDT          1I2   =>+0020020525
TIME-UPDT          1I2   =>+0000105000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[559]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[2] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO
POR-KEY            1X20  =>POR_KEY
VEND-NO            1I2   =>+0000000009
QTY-ORD            1I2   =>+0000000009
DATE-ORD           1I2   =>+0000000009
DATE-PROM          1I2   =>+0000000009
QTY-RECD           1I2   =>+0000000009
DATE-LAST          1I2   =>+0000000009
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
COMMENT            1X30  =>IMAXSOFT
UNIT-COST          1I4   =>000000000000000009
USER               1X8   =>IMAXSOFT
DATE-UPDT          1I2   =>+0020020526
TIME-UPDT          1I2   =>+0000141414
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[557]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO         1X36  =>CC_PNLC_PO_1
POR-KEY            1X20  =>POR_KEY_1
VEND-NO            1I2   =>+0000000001
QTY-ORD            1I2   =>+0000000888
```

```
DATE-ORD           1I2   =>+0000000001
DATE-PROM          1I2   =>+0000000001
QTY-RECD           1I2   =>+0000000666
DATE-LAST          1I2   =>+0000000001
REV-LOT            1X10  =>REV_LOT
REQ-NO             1X16  =>REQ_NO
COMMENT            1X30  =>COMMENT_IMS
UNIT-COST          1I4   =>0000000000000000001
USER               1X8   =>IMAXSOFT
DATE-UPDT          1I2   =>+0020020525
TIME-UPDT          1I2   =>+0000105000
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[555]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBCLOSE,0,,1]
DBCLOSE,INVENT.DATA;[8224],[-1],1,db_status[1]=0
=>> Repeat[1] TICommand[UNLOADTI]
```

3. OTDRV Program: generates ORACLE schema, unloads data and generates ORACLE table load scripts.

```
OpenTURBO*Pro OTDRV <A.01.03> iMaxsoft Corp. Copyright 2002.

iMAXSOFT/OpenTURBO iMAXSOFT Corp. Copyright 1993-, All Rights Reserved.
License No. 000000                          DEMO                          [2002/10/31]

run otdrv.bin;info='-d -ti -v -t -r -l -o -w -s -c -a -x -recnum -b'

  -dDBNAME      TurboIMAGE Database Name
                (Mode=4, Password=;) - for rootfile access
  -tiTIFile     This option is -ti, then the TIFileName
                TIFile is created by TILOAD program, it is
                OpenTURBO's rootfile. You use this option only
                for TRANCODE=20 and without -x.
  -vRWFile      Reserve Word File Name - this file is RDBMS
                specific, such as RESERVE.ORACLE is a default
                file name for ORACLE.
  -tTRANCODE    1   = Gen RDBSchema Without Automatic Dataset
                2   = Gen RDBSchema With Automatic Dataset
                20  = Gen RDBLoader ScriptFile and Unload
                      TurboIMAGE Data from one or more Datasets
                120 = Gen RDBLoader Script File Only
                      No TurboIMAGE Data Unload
  -rGrp.Acct    Unload Datafile's Group and Account; the Default
                is (OTDATA.HomeAcct)
  -lGrp.Acct    RDBLoader Scriptfile's Group and Account; the
                Default is (OTSCRIPT.HomeAcct)
  -oFile        For TRANCODE 1 and 2, it is the RDBSchema
                Filename and it must be Fully Qualified as
                <File.Grp.Acct>; the Default Filename is
                (OUTPUT.HomeGrp.HomeAcct)
  -oGrp.Acct    For TRANCODE 20 and 120, it is the Group and
                Account for storing the Working Copy of the
                TurboIMAGE Dataset File; the Default is
                (OTCOPY.HomeAcct)
  -wGrp.Acct    Temporary Working Group and Account; the Default
                is (OTWORK.HomeAcct)
  -sSET         TurboIMAGE Dataset Name or Number
  -s@           All Sets of the specified TurboIMAGE Database
  -s@A          All A-Sets of the specified TurboIMAGE Database
  -s@M          All M-Sets of the specified TurboIMAGE Database
  -s@D          All D-Sets of the specified TurboIMAGE Database
  -c            Preserve Detail Dataset Path Chronological
                Order (IMAXSOFT13_PATH_nn)
  -a            ASCII Transfer (default is BINARY)
  -x            Access TurboIMAGE in exclusive mode
  -recnum       Use TurboIMAGE RecNum for IMAXSOFT13_SEQ_NO
                which is a UNIQUE PRIMARY INDEX created by
                OpenTURBO, for Detail Dataset Only
  -bSQL_SERVER For SQL_SERVER Database, Default is ORACLE
                SQL_SERVER supports ASCII Transfer Only

  Note 1: all generated file names are the same as TurboIMAGE
          dataset MPE-FileName (such as: SAMPLE01, SAMPLE33).
  Note 2: the -oFile is for TRANCODE 1 and 2, it is a File
          Name not a Group Name.
  Note 3: you must NEWGROUP all working groups before running
          this program. Default groups are OTDATA, OTSCRIPT,
          OTCOPY and OTWORK.
```

-d TurboIMAGE database name, it can be qualified with group and account, as long as your login has proper access capability. During the process, we highly recommend that you DO NOT access the database concurrently.

-ti OpenTURBO root-file name, it is also known as TIFile.

-v Relational database reserved word file name.

-t OTDRV transaction code, OTDRV is a multi-function program, it allows you to generate ORACLE schema only, unload data and its corresponding ORACLE table load script, or generates ORACLE table load script only.

Transaction 1, generates ORACLE schema without TurboIMAGE automatic datasets.

Transaction 2, generates ORACLE schema with TurboIMAGE automatic datasets.

Transaction 20, unloads data from TurboIMAGE and generates ORACLE table load scripts.

Transaction 120, generates ORACLE table load scripts only.

-r The group and account for the unload data files. The default is OTDATA in your current login account.

The actual unload data file name is the same as the dataset file name.

-l The group and account for the ORACLE table load script files. The default is OTSCRIPT in your current login account.

The actual ORACLE table load script file name is the same as the dataset file name.

-o For transaction 1 and 2, it is the ORACLE schema file name, you must provide a fully qualified name, which is filename.group.account. The default is OUTPUT.home_group.home_account.

For transaction 20 and 120, it is the group and account for the working copy of TurboIMAGE detail datasets. The default is OTCOPY in your current login account. The actual working copy file name is the same as the dataset file name.

-w The group and account for OpenTURBO working area. The default is OTWORK in your current login account.

-s Specifies the dataset name that is going to be migrated. You may specify a specific dataset name, @A for all automatic datasets, @M for all manual master datasets, @D for all detail datasets, and @ for all datasets.

-c For transaction 1, 2, and 20 only, option to preserve detail dataset path chronological order, it allows you to preserve and migrate detail dataset path chronological order over to ORACLE database, so your mode 5 and mode 6 DBGET will generate results based upon the FIFO (mode 5) and LIFO (mode 6) order. The chronological order is preserved not only at migration time, it is also maintained thereafter via OpenTURBO emulator.

-a For transaction 20 only, option to unload data in text format, you be able to access it via editor. It is a good data clean-up tool. The default is BINARY.

We recommend to use BINARY for ORACLE files, it occupies less spaces and less chance to hit MPE file size limit.

For SQL SERVER, the TEXT is the only supported format.

-x Accesses TurboIMAGE in exclusive mode, even though, OTDRV does allow concurrent access of the TurboIMAGE while data loading, but it is highly recommend that you grant OTDRV in exclusive mode in all time.

```
      Transaction 20, if –x is specified, then OTDRV will access the
      TurboIMAGE in exclusive mode, otherwise, OTDRV will access the
      OpenTURBO root-file instead.

      All other transactions, OTDRV will access the TurboIMAGE in
      exclusive mode always.

   -recnum  Uses the TurboIMAGE internal record number as the unique
      sequence number in the migrated ORACLE database, it is
      particularly handy for parallel testing of your applications. DOOR
      guarantees your TurboIMAGE database and ORACLE database are 100%
      mirrored, including the internal record numbers.

      In a regular mode, non -recnum, any updates (DBPUT, DBDELETE and
      DBUPDATE) after the initial migration set different record number
      in TurboIMAGE versus ORACLE, which causes different access order
      for mode 2 and mode 3 serial scan.

      This option is required, if you are using our DOOR replication
      tool for synchronizing TurboIMAGE and ORACLE databases in real-
      time mode.

      A DOOR control table, DOOR_SYNC, is also created.

   -b SQL_SERVER for database SQL SERVER and default is ORACLE.

   -test Internal use only.

Examples:

   Generates ORACLE schema without automatic datasets:

      purge smusic1.otschema
      otdrv.bin;info='-dmusic.tidata.imaxsoft -t1 –osmusci1.otschema -c'

   Generates ORACLE schema with automatic datasets:

      purge smusic1.otschema
      otdrv.bin;info='-dmusic.tidata.imaxsoft –t2 –osmusci1.otschema -c'

      Sample ORACLE schema:

CREATE TABLE OT.TISET (
   TIBASE_NAME          VARCHAR2(26) NOT NULL,
   TABLE_NAME          VARCHAR2(256) NOT NULL,
   TISET_NAME           VARCHAR2(16) NOT NULL,
   TISET_NO               NUMBER(10) NOT NULL,
   TISET_TYPE           VARCHAR2(01) NOT NULL,
   LAST_SEQ_NO            NUMBER(20) NOT NULL,
   TIMESTAMP            VARCHAR2(14) NOT NULL
   );

CREATE TABLE OT.TISET_FIELD (
   TIBASE_NAME          VARCHAR2(26) NOT NULL,
   TISET_NO               NUMBER(10) NOT NULL,
   TIFIELD_SEQ_NO         NUMBER(10) NOT NULL,
   COLUMN_NAME         VARCHAR2(256) NOT NULL,
   TIITEM_NAME          VARCHAR2(20) NOT NULL,
   TIITEM_NO              NUMBER(10) NOT NULL,
   TIITEM_COUNT           NUMBER(10) NOT NULL,
   TIITEM_TYPE          VARCHAR2(01) NOT NULL,
   TIITEM_LENGTH          NUMBER(10) NOT NULL,
   COLUMN_TYPE            NUMBER(10) NOT NULL,
   COLUMN_LENGTH          NUMBER(10) NOT NULL,
   TIMESTAMP            VARCHAR2(14) NOT NULL
   );

CREATE TABLE OT.TISET_KEY (
   TIBASE_NAME          VARCHAR2(26) NOT NULL,
   TISET_NO               NUMBER(10) NOT NULL,
```

```
   TIKEY_SEQ_NO          NUMBER(10) NOT NULL,
   GEN_PATHCOLUMN      VARCHAR2( 1) NOT NULL,
   TIMASTERSET_NO        NUMBER(10) NOT NULL,
   TIINDEX_ITEM_NO       NUMBER(10) NOT NULL,
   COLUMN_NAME         VARCHAR2(256) NOT NULL,
   TIINDEX_ITEM_NAME   VARCHAR2(20) NOT NULL,
   TISORT_ITEM_NO        NUMBER(10) NOT NULL,
   TIMESTAMP           VARCHAR2(14) NOT NULL
   );

/************************************/
/* TIBASE:MUSIC.TIDATA.IMAXSOFT      */
/************************************/

/*************************/
/* TISET:ALBUMS           */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','ALBUMS','ALBUMS',1,'M',8,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',1,1,'N',1,1,'ALBUMCODE','ALBUMCODE',0,'20020716101418');

INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,1,'ALBUMCODE','ALBUMCODE',1,1,'I',2,1,10,'2002071610141
8');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,2,'ALBUMTITLE','ALBUMTITLE',3,1,'X',40,3,40,'2002071610
1418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,3,'MEDIUM','MEDIUM',12,1,'X',2,3,2,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,4,'ALBUMCOST','ALBUMCOST',2,1,'P',8,1,7,'20020716101418
');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,5,'RECORDINGCO','RECORDINGCO',15,1,'X',16,3,16,'2002071
6101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,6,'DATERECORDED','DATERECORDED',9,1,'X',16,3,16,'200207
16101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,7,'MFGCODE','MFGCODE',13,1,'X',40,3,40,'20020716101418'
);
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,8,'COMMENT_IMS','COMMENT',7,1,'X',80,3,80,'200207161014
18');

/*************************/
/* TISET:COMPOSERS        */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','COMPOSERS','COMPOSERS',2,'M',4,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',2,1,'N',2,8,'COMPOSERNAME','COMPOSERNAME',0,'200207161014
18');

INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,1,'COMPOSERNAME','COMPOSERNAME',8,1,'X',16,3,16,'200207
16101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,2,'BIRTH','BIRTH',5,1,'X',16,3,16,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,3,'DEATH','DEATH',10,1,'X',16,3,16,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,4,'BIRTHPLACE','BIRTHPLACE',6,1,'X',40,3,40,'2002071610
1418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,5,'COMMENT_IMS','COMMENT',7,1,'X',80,3,80,'200207161014
18');

/*************************/
/* TISET:SELECTIONS-A     */
/*************************/
INSERT INTO OT.TISET VALUES ('MUSIC.TIDATA.IMAXSOFT','SELECTIONS_A','SELECTIONS-
A',3,'A',8,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',3,1,'N',3,16,'SELECTIONNAME','SELECTIONNAME',0,'200207161
01418');
```

```
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',3,1,'SELECTIONNAME','SELECTIONNAME',16,1,'X',40,3,40,'200
20716101418');

/*************************/
/* TISET:SELECTIONS       */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','SELECTIONS','SELECTIONS',4,'D',8,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',4,1,'Y',1,1,'ALBUMCODE','ALBUMCODE',0,'20020716101418');
INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',4,2,'Y',3,16,'SELECTIONNAME','SELECTIONNAME',0,'200207161
01418');
INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',4,3,'Y',2,8,'COMPOSERNAME','COMPOSERNAME',0,'200207161014
18');

INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,1,'ALBUMCODE','ALBUMCODE',1,1,'I',2,1,10,'2002071610141
8');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,2,'SELECTIONNAME','SELECTIONNAME',16,1,'X',40,3,40,'200
20716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,3,'COMPOSERNAME','COMPOSERNAME',8,1,'X',16,3,16,'200207
16101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,4,'TIMING','TIMING',18,1,'X',16,3,16,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,5,'PERFORMERS','PERFORMERS',14,1,'X',40,3,40,'200207161
01418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,6,'COMMENT_IMS','COMMENT',7,1,'X',80,3,80,'200207161014
18');

/*************************/
/* TISET:LOG              */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','LOG','LOG',5,'D',6,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',5,1,'Y',1,1,'ALBUMCODE','ALBUMCODE',0,'20020716101418');
INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',5,2,'Y',3,16,'SELECTIONNAME','SELECTIONNAME',0,'200207161
01418');

INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,1,'ALBUMCODE','ALBUMCODE',1,1,'I',2,1,10,'2002071610141
8');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,2,'SELECTIONNAME','SELECTIONNAME',16,1,'X',40,3,40,'200
20716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,3,'STARTTIME','STARTTIME',17,1,'X',16,3,16,'20020716101
418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,4,'ENDTIME','ENDTIME',11,1,'X',16,3,16,'20020716101418'
);
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,5,'ANNOUNCER','ANNOUNCER',4,1,'X',40,3,40,'200207161014
18');

/***********************************/
/* TIBASE:MUSIC_TIDATA_IMAXSOFT      */
/***********************************/

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.ALBUMS (
   ALBUMCODE                    NUMBER(  10) NOT NULL, /* I[2] */
   ALBUMTITLE                VARCHAR2(  40) NOT NULL, /* X[40] */
   MEDIUM                    VARCHAR2(   2) NOT NULL, /* X[2] */
   ALBUMCOST                    NUMBER(   7) NOT NULL, /* P[8] */
   RECORDINGCO               VARCHAR2(  16) NOT NULL, /* X[16] */
   DATERECORDED              VARCHAR2(  16) NOT NULL, /* X[16] */
   MFGCODE                   VARCHAR2(  40) NOT NULL, /* X[40] */
   COMMENT_IMS               VARCHAR2(  80) NOT NULL, /* X[80] */
   IMAXSOFT13_SEQ_NO            NUMBER(  20) NOT NULL
   );
   /* 8 ROWS */
```

```
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_ALBUMS ON
MUSIC_TIDATA_IMAXSOFT.ALBUMS (ALBUMCODE);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_ALBUMS ON
MUSIC_TIDATA_IMAXSOFT.ALBUMS (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.COMPOSERS (
   COMPOSERNAME                VARCHAR2(  16) NOT NULL, /* X[16] */
   BIRTH                       VARCHAR2(  16) NOT NULL, /* X[16] */
   DEATH                       VARCHAR2(  16) NOT NULL, /* X[16] */
   BIRTHPLACE                  VARCHAR2(  40) NOT NULL, /* X[40] */
   COMMENT_IMS                 VARCHAR2(  80) NOT NULL, /* X[80] */
   IMAXSOFT13_SEQ_NO             NUMBER(  20) NOT NULL
   );
   /* 4 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_COMPOSERS ON
MUSIC_TIDATA_IMAXSOFT.COMPOSERS (COMPOSERNAME);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_COMPOSERS ON
MUSIC_TIDATA_IMAXSOFT.COMPOSERS (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (
   SELECTIONNAME               VARCHAR2(  40) NOT NULL, /* X[40] */
   IMAXSOFT13_SEQ_NO             NUMBER(  20) NOT NULL
   );
   /* 8 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_SELECTIONS_A ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (SELECTIONNAME);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS_A ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS (
   ALBUMCODE                     NUMBER(  10) NOT NULL, /* I[2] */
   SELECTIONNAME               VARCHAR2(  40) NOT NULL, /* X[40] */
   COMPOSERNAME                VARCHAR2(  16) NOT NULL, /* X[16] */
   TIMING                      VARCHAR2(  16) NOT NULL, /* X[16] */
   PERFORMERS                  VARCHAR2(  40) NOT NULL, /* X[40] */
   COMMENT_IMS                 VARCHAR2(  80) NOT NULL, /* X[80] */
   IMAXSOFT13_PATH_01            NUMBER(  10) NOT NULL,
   IMAXSOFT13_PATH_02            NUMBER(  10) NOT NULL,
   IMAXSOFT13_PATH_03            NUMBER(  10) NOT NULL,
   IMAXSOFT13_SEQ_NO             NUMBER(  20) NOT NULL
   );
   /* 8 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (ALBUMCODE, IMAXSOFT13_PATH_01);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (SELECTIONNAME, IMAXSOFT13_PATH_02);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (COMPOSERNAME, IMAXSOFT13_PATH_03);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I04_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.LOG (
   ALBUMCODE                     NUMBER(  10) NOT NULL, /* I[2] */
   SELECTIONNAME               VARCHAR2(  40) NOT NULL, /* X[40] */
   STARTTIME                   VARCHAR2(  16) NOT NULL, /* X[16] */
   ENDTIME                     VARCHAR2(  16) NOT NULL, /* X[16] */
   ANNOUNCER                   VARCHAR2(  40) NOT NULL, /* X[40] */
   IMAXSOFT13_PATH_01            NUMBER(  10) NOT NULL,
   IMAXSOFT13_PATH_02            NUMBER(  10) NOT NULL,
   IMAXSOFT13_SEQ_NO             NUMBER(  20) NOT NULL
   );
   /* 6 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_LOG ON MUSIC_TIDATA_IMAXSOFT.LOG
(ALBUMCODE, IMAXSOFT13_PATH_01);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_LOG ON MUSIC_TIDATA_IMAXSOFT.LOG
(SELECTIONNAME, IMAXSOFT13_PATH_02);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_LOG ON MUSIC_TIDATA_IMAXSOFT.LOG
(IMAXSOFT13_SEQ_NO);
```

```
CREATE TABLE MUSIC_TIDATA_IMAXSOFT.DOOR_SYNC  (
   DBNAME           CHAR(30) NOT NULL,
   LOGRECNUM        INTEGER  NOT NULL,
   LOGFILENAME      CHAR(30) NOT NULL
   );

INSERT INTO MUSIC_TIDATA_IMAXSOFT.DOOR_SYNC VALUES ('-',0,' ');

COMMIT WORK;
EXIT;
```

Unloads database DEMO and creates table load script:

```
setvar ltdbg1 0
setvar ltdbg2 0
setvar ltdbg3 0
setvar ltdbg4 0
setvar ltdbgout '$stdlist'
otdrv.bin;info= ' &
-dmusic.tidata.moulton &
-ti ti.otcmd.imaxsoft &
-v reserve.oracle &
-t20 &
-c   &
-s@'
```

1) Turn OTDRV internal debugging trace off (ltdbg1 through ltdbg4), if there is any errors prints to ltdbgout file which is set to $stdlist.
2) The OpenTURBO root-file for MUSIC is TI.OTCMD.IMAXSOFT.
3) Unloads all datasets into group OTDATA.IMAXSOFT.
4) Generates ORACLE table load scripts into group OTSCRIPT.IMAXSOFT.
5) Preserves and migrates TurboIMAGE detail dataset path chronological orders.
6) Binary data transfer by default.

TurboIMAGE Unload Dataset Files:

```
ACCOUNT=  IMAXSOFT    GROUP=  OTDATA

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

MUSIC01           103W  FB           8          8   1       16  1  5
MUSIC02            86W  FB           4          4   1       16  1  5
MUSIC03            22W  FB           8          8   1       16  1  5
MUSIC04           106W  FB           8          8   1       16  1  5
MUSIC05            64W  FB           6          6   1       16  1  7
```

ORACLE Table Loader Script Files:

```
ACCOUNT=  IMAXSOFT    GROUP=  OTSCRIPT

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

MUSIC01           256B  VA          15      90000   1      256  1  8
MUSIC02           256B  VA          12      90000   1      256  1  8
MUSIC03           256B  VA           8      90000   1      256  1  8
MUSIC04           256B  VA          16      90000   1      256  1  8
MUSIC05           256B  VA          14      90000   1      256  1  8
```

ORACLE Table Load Script File MUSIC04:

```
LOAD DATA
INFILE 'MUSIC04.DAT' "fix 212"
TRUNCATE
PRESERVE BLANKS
INTO TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS (
IMAXSOFT13_PATH_01     POSITION(0001:0004) INTEGER,
IMAXSOFT13_PATH_02     POSITION(0005:0008) INTEGER,
IMAXSOFT13_PATH_03     POSITION(0009:0012) INTEGER,
ALBUMCODE              POSITION(0013:0016) INTEGER,
SELECTIONNAME          POSITION(0017:0056) CHAR,
COMPOSERNAME           POSITION(0057:0072) CHAR,
TIMING                 POSITION(0073:0088) CHAR,
PERFORMERS             POSITION(0089:0128) CHAR,
COMMENT_IMS            POSITION(0129:0208) CHAR,
IMAXSOFT13_SEQ_NO      POSITION(0209:0212) INTEGER
)
```

GENFTP: generates ftp file transfer script to transfer all OpenTURBO generated files from HP3000 to HP9000.

The GENFTP program:

```
Generate FTP Script <A.01.03> iMaxsoft Corp.Copyright 2002.
run genftp.bin;info='-d -r -l -k -o -h -p -u'

   -dDBNAME:    TurboIMAGE Fully Qualified Database Name
   -rDAT   :    OpenTURBO Generated TurboImage Data File
                Group
   -lCLT   :    OpenTURBO Generated ORACLE Loader Script
                File Group
   -kSCHEMA:    OpenTURBO Generated ORACLE Schema File Group
   -oOUT   :    Generated FTP Job FileName
   -hHOST  :    Target HP-UX Host IP or DNS Name
   -pDir   :    Target HP-UX Directory for Transferred Files
   -uUSER  :    HP-UX Login User & Password (USER/PASSWD)
   -a      :    ASCII Mode Transfer Data (Default is Binary)
```

Note: All password are encrypted via program encrypt.

Sample run script:

```
genftp.bin;info= ' &
   -dMEMBRS &
   -rDMDAT &
   -lDMCLT &
   -kDMSCHEMA &
   -oJLDFTP &
   -h207.92.64.8 &
   -pdemo/data &
   -ulee/gzz0122'
```

4. TILOAD Program: generates OpenTURBO TurboIMAGE Internal root-file, so called TI root-file. TILOAD is also used for checking TIFile version and re-creating TurboIMAGE schema.

   TILOAD program:

```
   OpenTURBO TILOAD <A.01.03> iMaxsoft Corp. Copyright 2002.
   run tiload.bin;info='-d -t -r -v -m'
      -dDBName : TurboIMAGE Database Name
      -tTIFile : OpenTURBO Root-File Name
      -rRWFile : OpenTURBO Reserve Word File Name
      -v       : Check OpenTURBO Root-File Version
      -mOUTFile: Print OpenTURBO Root-File Schema to OUTFile
```

   Examples:

   TI root-file creation:

```
      tiload.bin;info='-dmusic.tidata.imaxsoft -tti.otcmd -
      rreserve.oracle'
```

   TI root-file version checking:

```
      tiload.bin;info='-dmusic.tidata.imaxsoft -tti.otcmd -
      rreserve.oracle –v'
```

   TurboIMAGE schema re-creation:

```
      tiload.bin;info='-dmusic.tidata.imaxsoft -tti.otcmd -
      rreserve.oracle –mOTMUSIC.TISCHEMA'

      DBName=MUSIC.TIDATA.IMAXSOFT; Version=A.01.03
      Process Start: 2002-07-16 18:07:11
      Process Stop:  2002-07-16 18:07:11
```

   OTMUSIC.TISCHEMA.IMAXSOFT File Printout:

```
TURBOIMAGE DATABASE  = MUSIC.TIDATA.IMAXSOFT;
TURBOIMAGE ITEM LIST :
      1 ALBUMCODE          1I2   ByteLen=   4
      2 ALBUMCOST          1P8   ByteLen=   4
      3 ALBUMTITLE         1X40  ByteLen=  40
      4 ANNOUNCER          1X40  ByteLen=  40
      5 BIRTH              1X16  ByteLen=  16
      6 BIRTHPLACE         1X40  ByteLen=  40
      7 COMMENT            1X80  ByteLen=  80
      8 COMPOSERNAME       1X16  ByteLen=  16
      9 DATERECORDED       1X16  ByteLen=  16
     10 DEATH              1X16  ByteLen=  16
     11 ENDTIME            1X16  ByteLen=  16
     12 MEDIUM             1X2   ByteLen=   2
     13 MFGCODE            1X40  ByteLen=  40
     14 PERFORMERS         1X40  ByteLen=  40
     15 RECORDINGCO        1X16  ByteLen=  16
     16 SELECTIONNAME      1X40  ByteLen=  40
     17 STARTTIME          1X16  ByteLen=  16
     18 TIMING             1X16  ByteLen=  16
TURBOIMAGE SET LIST  =
[M] 1   ALBUMS
   FIELD[  1]   1 ALBUMCODE          1I2   ByteLen=   4
   FIELD[  2]   3 ALBUMTITLE         1X40  ByteLen=  40
   FIELD[  3]  12 MEDIUM             1X2   ByteLen=   2
   FIELD[  4]   2 ALBUMCOST          1P8   ByteLen=   4
   FIELD[  5]  15 RECORDINGCO        1X16  ByteLen=  16
   FIELD[  6]   9 DATERECORDED       1X16  ByteLen=  16
   FIELD[  7]  13 MFGCODE            1X40  ByteLen=  40
   FIELD[  8]   7 COMMENT            1X80  ByteLen=  80
  *PKEY [  1]   1 ALBUMCODE
[M] 2   COMPOSERS
   FIELD[  1]   8 COMPOSERNAME       1X16  ByteLen=  16
   FIELD[  2]   5 BIRTH              1X16  ByteLen=  16
```

```
   FIELD[  3]  10 DEATH              1X16  ByteLen=  16
   FIELD[  4]   6 BIRTHPLACE         1X40  ByteLen=  40
   FIELD[  5]   7 COMMENT            1X80  ByteLen=  80
  *PKEY [  1]   8 COMPOSERNAME
[A] 3   SELECTIONS-A
   FIELD[  1]  16 SELECTIONNAME      1X40  ByteLen=  40
  *PKEY [  1]  16 SELECTIONNAME
[D] 4   SELECTIONS
   FIELD[  1]   1 ALBUMCODE          1I2   ByteLen=   4
   FIELD[  2]  16 SELECTIONNAME      1X40  ByteLen=  40
   FIELD[  3]   8 COMPOSERNAME       1X16  ByteLen=  16
   FIELD[  4]  18 TIMING             1X16  ByteLen=  16
   FIELD[  5]  14 PERFORMERS         1X40  ByteLen=  40
   FIELD[  6]   7 COMMENT            1X80  ByteLen=  80
  *PKEY [  0]   1 ALBUMCODE        MASTER=ALBUMS
  *PATH [  1]   1 ALBUMCODE        MASTER=ALBUMS           SORT=_NONE_
  *PATH [  2]  16 SELECTIONNAME    MASTER=SELECTIONS-A     SORT=_NONE_
  *PATH [  3]   8 COMPOSERNAME     MASTER=COMPOSERS        SORT=_NONE_
[D] 5   LOG
   FIELD[  1]   1 ALBUMCODE          1I2   ByteLen=   4
   FIELD[  2]  16 SELECTIONNAME      1X40  ByteLen=  40
   FIELD[  3]  17 STARTTIME          1X16  ByteLen=  16
   FIELD[  4]  11 ENDTIME            1X16  ByteLen=  16
   FIELD[  5]   4 ANNOUNCER          1X40  ByteLen=  40
  *PKEY [  0]   1 ALBUMCODE        MASTER=ALBUMS
  *PATH [  1]   1 ALBUMCODE        MASTER=ALBUMS           SORT=_NONE_
  *PATH [  2]  16 SELECTIONNAME    MASTER=SELECTIONS-A     SORT=_NONE_
```

```
TICOPY Program: makes a copy of OpenTURBO TI root-file with a
different TurboIMAGE name or upgrade/migrate TIFile version.

TICOPY program:

   OpenTURBO TICOPY <A.01.03> iMaxsoft Corp. Copyright 2002.
   run ticopy.bin;info='-d -t -r -n -o'
      -dDBName : CopyFrom Fully Qualified TurboIMAGE DBName
      -tTIFile : CopyFrom Fully Qualified TIFile Name
      -rRWFile : Reserve Word File Name
      -nDBName : CopyTo Fully Qualified TurboIMAGE DBName
                 Ignore means NO Change of TurboIMAGE DBName
      -oTIFile : CopyTo Fully Qualified TIFile Name
                 Ignore means TIFILE Version Migration

Examples:

   TI root-file copy:

      ticopy.bin;info='-dMUSIC.TIDATA.IMAXSOFT -tti.otcmd -
      rreserve.oracle -nMUSIC.TEST.IMAXSOFT -oMUSICTI.otcmd'

   TI root-file version upgrade:

      ticopy.bin;info='-dMUSIC.TIDATA.IMAXSOFT -tti.otcmd -
      rreserve.oracle'
```

5. TIDRV Program and Emulator's CONFIG File: OpenTURBO testing driver program. You may enter all TurboIMAGE API calls via its straight-forward syntax and verify the formatted results. All input commands and output results can be processed from files.

   TIDRV can be run in silent mode, which takes inputs from a command file and reports output to an output file; or run in interactive mode which uses SDTIN and STDOUT as the input and output files.

   TIDRV can be run in OpenTURBO mode as well as in TurboIMAGE mode. In OpenTURBO mode, it accesses ORACLE database either locally or remotely; in TurboIMAGE mode, it also can be configured to access TurboIMAGE database locally or remotely. The result formats are identical in both modes, you can diff them easily; TIDRV is the best tool used to verify data migration results, to perform progressive test, and to conduct performance benchmarking analysis.

   - On HP e3000, run TIDRV;XL="OTXL.A.IMAXSOFT" - accesses remote ORACLE database on HP9000.
   - On HP e3000, run TIDRV;XL="XL.PUB.SYS" - accesses local TurboIMAGE database.
   - On HP e3000, run TIDRV;XL="OTXL.A.IMAXSOFT";INFO="-turboimage" - accesses local TurboIMAGE database via OpenTURBO MPE/XL library.
   - On HP9000, run TIDRV with libot.sl - accesses local ORACLE database.
   - On HP9000, run TIDRV -turboimage with libot.sl - accesses remote TurboIMAGE database on HP e3000 via OpenTURBO HP-UX library.
   - You may also run DUAL-MODE from both HP e3000 and HP9000.

   You may use absolute file $_OTB_ROOT/conf/RESERVE.ORACLE directly, or copy RESERVE.ORACLE into your login MPE account, or use MPE file equation command :FILE RESERVE.ORACLE = to locate the file, or use -r to identify the RESERVE.ORACLE file.

   The CONFIG file: you may use environment variable OT_CONFIG to redirect it.

```
// This is used by TIDRV.BIN.IMAXSOFT - 2002-7-11 Lee Tsai
//
// For:  SYSTEM LEVEL CONFIGURATION FILE for OpenTURBO
//       This is used by HP/3000 Client SDK Version
// Date: 2002/04/18
//
// 515 is for HPUX and ORACLE
//
OT_TI_DBNAME = DEMO.DATA.IMAXSOFT
DEMO.DATA.IMAXSOFT {
DEMO.DATA {
DEMO {
   OT_IMAGEMODE          = ON
   OT_ROOT_FILE          = ti1.ti.ims
   OT_RESERVE_WORD_FILE = RESERVE.ORACLE.ims
   OT_ERROR_FILE         = OTERROR.ORACLE.ims
   OT_HOST               = 207.92.64.8
   OT_SERVICE            = 32601
   OT_OS_RDBMS           = 515
   OT_RDB_LOGON          = demo_data_imaxsoft/diqzio
   OT_RDB_SID            = v90
   OT_RDB_OWNER          = AMISYS
   OT_SDK_SERVER_PGM     = /users/lee/lbin/dbsvr
   OT_CIUPDATE           = ON
   OT_DUALMODE           = OFF
   TI_DUALMODE_HOST      = 207.92.64.66
   TI_DUALMODE_SERVICE   = 32608
   TI_DUALMODE_PGM       = DMDRV.BIN.IMAXSOFT
   OT_LOCKWAIT_CYCLE     = 60
   OT_TRX_THRESHOLD      = 10
```

```
//
// Database Level MODE 5 and MODE 6 Chain Get Controls:
//
// OT_IGNORE_CHAINSTATUS:
//    OT_IGNORE_CHAINSTATUS = ON: OpenTURBO ignores the OT_BULKCHAINSET
//    settings. OpenTURBO will perform either forward (MODE 5) or backward
//    (MODE 6) chain get without tracking chain_count, forward and backward
//    pointers. (It is the most efficient option, but status 3-4, 5-6,
//    7-8, and 9-10 are all 0.)
//
// OT_BULKCHAINGET:
//    OT_BULKCHAINGET = ON: Get as many rows as it can fit in a 30000 bytes
//    buffer - ONE DIRECTION ONLY, MODE 5 forward only, MODE 6 backward only.
//    OT_BULKCHAINGET = OFF: Get one row at a time, you may go forward and
//    backward by using the status 7-8 and 9-10 backward and forward record
//    pointers (100% emulates DBGET())
//
// OT_IGNORE_DBPUTSTATUS:
//    OT_IGNORE_DBPUTSTATUS = ON: OpenTURBO returns no record number after
//    successful DBPUT. This is a performance tuning factor.
//
// OT_USE_IMAGERECNUM:
//    OT_USE_IMAGERECNUM = ON: This is for DUAL-MODE only. OpenTURBO will perform
//    TurboIMAGE DBPUT first and then use the returned RecNum for ORACLE INSERT
//    statement. This is to ensure that the ORALCE is exact mirror of TurboIMAGE
//    database.
//
// OT_CHRONOLOGICAL:
//    OT_CHRONOLOGICAL = ON - OpenTURBO will sort the path via OpenTURBO
//    internal chronological sequence number IMAXSOFT13_PATH_nn; ASC for
//    MODE 5, DESC for MODE 6.
//
// OT_IMAGEMODE:
//    OT_IMAGEMODE = ON: OpenTURBO accesses the specific database in TurboIMAGE
//    native mode. OpenTURBO ignores all other settings but TI_DUALMODE_HOST,
//    TI_DUALMODE_SERVICE and TI_DUALMODE_PGM, if you intend to access TurboIMAGE
//    database remotely from HP9000.
//
// OT_CIUPDATE:
//    OT_CIUPDATE = ON - OpenTURBO will set IMAXSOFT13_PATH_nn = -888
//    to inform OpenTURBO Server program to trigger the appropriate
//    update procedure
//
// OT_DUALMODE:
//    OT_DUALMODE = ON - OpenTURBO will perform TurboIMAGE calls in
//    dual mode, OpenTURBO will execute standard TurboIMAGE first, then
//    emulated OpenTURBO call and compare the returned buffer as well as
//    status[1] - status[10]
//
// OT_LOCKWAIT_CYCLE:
//    Specify the number of retries allowed for unconditional lock request
//    (mode 1, 3, and 5). The default is 0, wait forever.
//
OT_DETAILSETNAME = @ [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = ON
   OT_CHRONOLOGICAL       = ON
]
}
```

Footnotes:

1) If OT_RDB_OWNER is used, then all DBOPEN to this specific database uses the OT_RDB_LOGON as the ORACLE logon instead of the mapped ORACLE user (dbname_group_account_classID).
2) You may specify ORACLE SID in the OT_RDB_LOGON string, such as v90:demo_data_imaxsoft/digzio.
3) You may also specify ORACLE SQLNet in the OT_RDB_LOGON string, such as demo_data_imaxsoft/digzeo@oraSVR.
4) OT_RDB_SID is reserved for A.03.00.

Database Level Configuration:

- DBNAME.GROUP.ACCOUNT, DBNAME.GROUP and DBNAME point to the same physical TurboIMAGE database.

- OT_ROOT_FILE points to the root file created by TILOAD. No environment variable is allowed in the file name, such as $HOME/db/DBNAMEti.ti gets error 'TIFile cannot be found'.
- OT_RESERVE_WORD_FILE points to the file that contains relational database reserve words and conversion suffix.
- OT_ERROR_FILE points to the file that contains the TurboIMAGE errors and messages for OpenTURBO.
- OT_HOST points to the machine where your ORACLE instance is logical (SQLNET) or physical located.
- OT_SERVICE directs OpenTURBO to the appropriate listener daemon on the OT_HOST.
- OT_DB_RDBMS specifies the combination of OS and RDBMS, value 515 is ORACLE on HP-UX, and 712 is ORACLE on INTEL-LINUX.
- OT_RDB_LOGON contains TurboIMAGE Class 64 and Password ';' equivalent relational database logon, the format is user/password, and the password is in its encrypted format, use ENCRYPT to generate your encrypted password.
- OT_SDK_SERVER_PRG contains the OpenTURBO Core Server Program name, /opt/imaxsoft/otb/lbin/dbsvr.
- OT_CIUPDATE denotes whether UPDATE Critical Item option is OT_CIUPDATE denotes whether UPDATE Critical Item option is allowed or not; this option is NULL, if you didn't migrate your detail set paths and associated automatic sets.
  To be able to use OT_CIUPDATE=ON correctly,
  1) you need to unload your detail dataset with option –c (OTDRV),
  2) if DOOR is used, you need to generate mapfile with option –c (DOORMAP),
  3) if DOOR is used, you need to start shooter with option –c (SHOOTOT),
  4) set OT_CHRONOLOGICAL = ON for the entire TurboIMAGE database or specific detail dataset.
- The OT_CIUPDATE = ON and OT_CHRONOLOGICAL = ON sets IMAXSOFT_PATH_nn = -888 in DBUPDATE, which triggers ORACLE update trigger to perform Automatic set maintenance procedure.
- OT_IMAGEMODE = ON, enables OpenTURBO to access TurboIMAGE in native mode. If your apps run on HP-UX, then you need to specify TI_DUALMODE_HOST, TI_DUALMODE_SERVICE and TI_DUALMODE_PRG for remote TurboIMAGE access.
  The option is also used in conjunction with OT_DUALMODE for specifying the primary database.
- OT_DUALMODE enables OpenTURBO concurrent dual-access of both TurboIMAGE and ORACLE and reports discrepancy.

OS	OT_IMAGEMODE	OT_DUALMODE	Comment
MPE/XL	ON	OFF	TurboIMAGE only
MPE/XL	-	ON	ORACLE primary TurboIMAGE secondary
MPE/XL	OFF	OFF	ORACLE only
HP-UX	ON	OFF	TurboIMAGE only
HP-UX	-	ON	ORACLE primary TurboIMAGE secondary
HP-UX	OFF	OFF	ORACLE only

- TI_DUALMODE_HOST, TI_DUALMODE_SERVICE, and TI_DUALMODE_PGM are used to connect to TurboIMAGE on HP/3000 from your HP-UX migrated programs. You must start the Listener on the HP/3000 before running your program on the HP-UX in DUAL-MODE.
- OT_LOCKWAIT_CYCLE controls the number of wait cycles, when your application uses unconditional LOCK option, each wail cycle is 2 seconds and default cycle is wait forever which is equivalent to OT_LOCKWAIT_CYCLE = 0.

- OT_TRX_THRESHOLD controls the SQL transaction run-time trace when you use ltxldbg (HP e3000) or libotdbg (HP-UX) libraries and trigger OpenTURBO debugging facility via 'SETVAR LTDBG7 1' or 'EXPORT LTDBG7=1'. Default threshold is 10 seconds. It traps any transaction that takes longer than threshold limit and reports it for performance analysis.

Dataset Level Configuration:

- OT_DETAILSETNAME specifies the name of a detail dataset, @ for all detail datasets.
- OT_IGNORE_CHAINSTATUS = ON, OpenTURBO ignores the OT_BULKCHAINGET. OpenTURBO will allow either forward (mode 5) or backward (mode 6) chain get without tracking chain count and forward and backward pointers. If one of your application does only one direction chain get and does not use status words 5-10, the you should use this option to significant improve its through-put and performance.
- OT_BULKCHAINGET = ON, OpenTURBO will trigger bulk fetch and get as many rows as they can to fit internal buffer, whereas in the standard mode, OpenTURBO gets one row at a time;
  OT_BULKCHAINGET option supports one direction chain get only, either DBGET Mode 5 forward or DBGET Mode 6 backward, but not both. Ignored, if OT_IGNORE_CHAINSTATUS = ON.
- OT_CHRONOLOGICAL = ON, OpenTURBO will sort IMAXSOFT13_PATH_nn in ascending order for DBGET Mode 5 and in descending order for DBGET Mode 6.
  In the DBPUT, this option will force OpenTURBO to maintain the chronological order of each IMAXSOFT13_PATH_nn. CIUPDATE is applicable only for DBUPDATE, whereas the OT_CHRONOLOGICAL is for DBPUT, DBUPDATE and DBDELETE.
- OT_IGNORE_DBPUTSTATUS = ON, OpenTURBO returns no record number after successful DBPUT, this is strictly for performance.
- OT_USE_IMAGERECNUM = ON, this is for DUAL-MODE only. OpenTURBO will DBPUT to TurboIMAGE first, then use the returned record number for ORACLE INSERT statement. This is to assure an exact mirror of TurboIMAGE and ORACLE.

The TIDRV program:

```
OpenTURBO TIDRV <A.01.04> iMaxsoft Corp. Copyright 2002.

run tidrv.bin;info='-r -i -o -map -turboimage'
   -rRWFile    OpenTURBO Reserve Word File Name
   -iINPUT     TurboIMAGE Command Input File Name
   -oOUTPUT    Output File Name
   -map        Print TurboIMAGE Schema
   -turboimage Access TurboIMAGE Directly via OpenTURBO Library

tidrv commands:
 1) // or /*          for comment lines
    &                 for command continuation
    ''                for character string
    \                 for de-reference \
    ,                 for field separator
    COMMENT ON        to turn on comment block
    COMMENT OFF       to turn off comment block
    PRINT ON          to turn on print-to-file
    PRINT OFF         to turn off print-to-file
    LOADTI,TIFileName to load TI from TIFileName
    UNLOADTI          to unload and free TI (OBSOLETE)
    USETI,TIFile_ID   set active TIFile for subsequent TIDRV
                      command syntax checking.
                      The first  LOADTI assigns TIFile_ID = 0,
                      the second LOADTI assigns TIFile_ID = 1,
                      and so on, up to total of 64.
    REPEAT n          repeat subsequent DBcall n times
    DEFINE short var value
    DEFINE int   var value
    DEBUGOUT filename set debugging output filename
    DEBUGn ON/OFF     n = 1 through 31
    EXIT              end program
```

```
 2) All TurboIMAGE DBCalls:
    DBCLOSE,baseID,dset,mode,status
    DBCONTROL,baseID,QUALIFIER=,mode,status
    DBDELETE,baseID,dset,mode,status
    DBERROR,status,DATA=,textlen
    DBEXPLAIN,status
    DBFIND,baseID,dset,mode,status,ITEM=,ARG=
    DBGET,baseID,dset,mode,status,LIST=,DATA=,ARG=
    DBINFO,baseID,QUALIFIER=,mode,status,DATA=
    DBLOCK,baseID,QUALIFIER=,mode,status
    DBOPEN,base,password,mode,status
    DBPUT,baseID,dset,mode,status,LIST=,DATA=
    DBUNLOCK,baseID,dset,mode,status
    DBUPDATE,baseID,dset,mode,status,LIST=,DATA=
    DBXBEGIN,BASE=id/BASELIST=id:id:id,mode
    DBXEND,BASE=id/BASELIST=id:id:id,mode
    DBXUNDO,BASE=id/BASELIST=id:id:id,mode
 ** Will support in the next release:
    DBBEGIN,[baseID,baselist],DATA=,mode,status,textlen
    DBEND,[baseID,baselist],DATA=,mode,status,textlen
    DBMEMO,baseID,DATA=,mode,status,textlen
```

Run options:

1) -r OpenTURBO Reserve Word File Name. You may use absolute file $_OTB_ROOT/conf/RESERVE.ORACLE directly, or copy RESERVE.ORACLE into your login MPE account, or use MPE file equation command :FILE RESERVE.ORACLE = to locate the file, or use -r to identify the RESERVE.ORACLE file.
2) -map prints TurboIMAGE schema at front first.
3) -i specifies the input command file name; on HP3000, use $stdin for interactive mode; on HP9000, use - for interactive mode.
4) -o specifies the output result file name; on HP3000, use $stdlist for terminal output; on HP9000, use - for terminal output.

TIDRV control commands:

1) Use // or /* for comment line
2) Use COMMENT ON and COMMENT OFF for comment block
3) Use & at end of each command line for denoting command continuation
4) All value must be embedded in single quote ‘’
5) Use \ for de-reference special character, such as \
6) Use , for parameters separator
7) PRINT ON and PRINT OFF to turn on and off print-result-to-file option
8) DEFINE is used to declare variables, currently we only support short and int, which are 16-bit and 32-bit interger
9) REPEAT n, executes the immediate followed DBCall n times, one call only
10) DEBUGOUT filename, specifies the remote debugging file name (on HP9000)
11) DEBUGn ON and DEBUGn OFF to turn on and off debugging level from 0 through 31; currently supported levels are:

    DEBUG0: Serious Error (no need to turn on)
    DEBUG1: OpenTURBO core level trace
    DEBUG2: OpenTURBO reserved word
    DEBUG3: OpenTURBO mapped error message (TurboIMAGE)
    DEBUG4: OpenTURBO emulator level trace
    DEBUG5: OpenTURBO client SQL statement and CURSOR POOL trace
    DEBUG6: OpenTURBO DULA MODE diff results
    DEBUG7: OpenTURBO transaction performance trace
    DEBUG13: OpenTURBO call pattern analyzer
    DEBUG17: Network traffic dump in hex and text
    DEBUG18: Network socket trace

```
     DEBUG19: Network Net/IPC and TCP/IP trace
     DEBUG27: SQL statement dump
     DEBUG28: SQL error analyzer
     DEBUG29: malloc() and free() trace

12)  LOADTI,ti-filename and UNLOADTI (obsolete); loads the OpenTURBO
     root-file into memory for DBCalls syntax and semantic checking
13)  USETI,tifile-id; sets the active OpenTURBO root-file (tifile)
     for subsequent DBCalls syntax and semantic checking; the first
     LOADTI gets 0 for tifile-id, the second LOADTI gets 1 for
     tifile-id, and so on till 63
14)  EXIT ends TIDRV program

TIDRV TurboIMAGE calls:

1)   DBOPEN , TurboIMAGE_Database_Name; , Password; , Mode

     o  The first  DBOPEN's baseID = 0
     o  The second DBOPEN's baseID = 1
     o  The third  DBOPEN's baseID = 2
     o  . . . and so on

2)   DBCLOSE , baseID , Dataset-Name | Dataset-Number | None , Mode

     o  The baseID is the number associated to the DBOPEN
     o  None means nothing in between two commas, such as ,,
     o  The Dataset-Name is 16 characters long or terminated by
        either blank or semi-colon, such as MEMBERDETL;
     o  The Dataset-Number is number only, such as 24

3)   DBFIND , baseID , Dataset-Name | Dataset-Number , Mode ,
     ITEM='Item-Name | Item-Number' , ARG=Defined-Variable | 'Value'

     o  The ITEM= is TIDRV's key word and is part of comamnd syntax
     o  The Item-Name is the key item, it can be 16 characters long
        or terminated by either blank or semi-colon, such as MBRNO
     o  The Item-Number is number only, such as 5
     o  The ARG= is TIDRV's key word and is part of command syntax
     o  The Defined-Variable is declared via TIDRV DEFINE command
     o  The Value can be a true value, value with wildcard, or the
        standard argument as specified in TurboIMAGE manual page 180
     o  OpenTURBO A.01.00 support all modes but 10, which has been
        implemented but has not been certified by TPI vendors yet.

4)   DBGET , baseID , Dataset-Name | Dataset-Number , Mode ,
     LIST='Item-Name List | Item-Number List | Special List' ,
     ARG=Defined-Variable | 'Value'

     o  The LIST= is TIDRV's key word and is part of command syntax
     o  Item-Name List is a list of item names separated by comma,
        such as MBRNO, MBRNAME, . . .
     o  Item-Number List is a list of item numbers separated by
        comma, such as 24, 5, . . .
     o  Special List has specific meaning, such as @; means all
        items, *; means same as previous DBCall List, and so on
     o  ARG= is used for Manual Master calculated get by key value or
        direct get by record number

5)   DBERROR is part of DBEXPLAIN, use DBEXPLAIN instead

6)   DBEXPLAIN , baseID

7)   DBCONTROL , baseID , QUALIFIER='' , Mode
```

- The QUALIFIER= is TIDRV's key word and is part of command syntax
- Supported modes:

```
Mode  5:   Enables the critical item update option
Mode  6:   Disables the critical item update option
Mode  7:   Allows Dynamic Multiple Database Transaction
Mode 88:   Turns ON/OFF a remote debugger level, use the first
           half-word of QUALIFIER= for the debugging level (0
           through 31) and the second half-word of QUALIFIER=
           for the ON(1) and OFF(0) switch
Mode 89:   Sets the remote debugger file name, such as
           QUALIFIER='debugger file name'
```

8) DBINFO , baseID , QUALIFIER='' , Mode

- Refer to TurboIMAGE manual for QUALIFIER=, page 190

9) DBLOCK , baseID , QUALIFIER='' , Mode

- Refer to TurboIMAGE manual for QUALIFIER=, page 207 shows the detail format of the lock descriptor
- CLIENT-LOCK-MANAGER is responsible for checking and enforcing process related lock rules
- SERVER-LOCK-MANAGER is responsible for checking and enforcing cross-process lock rules

10) DBUNLOCK , baseID , None , Mode

11) DBPUT , baseID , Dataset-Name | Dataset-Number , Mode , LIST='' , DATA=''

12) DBDELETE , baseID , Dataset-Name | Dataset-Number , Mode

13) DBUPDATE , baseID , Dataset-Name | Dataset-Number , Mode , LIST='' , DATA=''

14) DBXBEGIN , baseID | baseID:baseID:baseID:. . , Mode

15) DBXEND , baseID | baseID:baseID:baseID:. . , Mode

16) DBXUNDO , baseID | baseID:baseID:baseID:. . , Mode

17) DBBEGIN – next version

18) DBEND – next version

19) DBMEMO – next version

Example:

HP3000:

Run script:

```
parm xl='NOOT'
setvar xlflag '!xl'
setvar ltdbg1 0
setvar ltdbg2 0
setvar ltdbg3 0
setvar ltdbg4 1
setvar ltdbg5 1
setvar ltdbg6 1
setvar ltdbg17 0
setvar ltdbg18 0
setvar ltdbg19 0
setvar ltdbgout '$stdlist'
```

```
purge outtrx1.output
if (xlflag = 'OT') then
   file config=config.lee.ims
   run tidrv.bin;xl='otxldbg.a,tidrv.a,ltxl.pub.imaxsoft'; &
   info='-iintrx1.lee -oouttrx1.output'
else
   run tidrv.bin;xl='tidrv.a,ltxl.pub.imaxsoft'; &
   info='-iintrx1.lee -oouttrx1.output'
endif
```

1) OT triggers TIDRV to run in OpenTURBO emulator mode
2) NOOT triggers TIDRV to run in normal TurboIMAGE mode
3) The input command file name is intrx1
4) The output result file name is outtrx1

Input command file:

```
LOADTI,ti1.ti
// ** DBPUT to Detail Dataset - PURCHASE
DBOPEN,INVENT.DATA.MOULTON;,FAVOR;,3
// DBOPEN,INVENT.DATA.MOULTON;,;,1
//
COMMENT ON
DEBUGOUT /users/lee/tmp/lee.dbg
DEBUG19 ON
DEBUG18 ON
DEBUG17 ON
DEBUG27 ON
DEBUG28 ON
COMMENT OFF
//
DBCONTROL,0,,7
//
//         ==========>> Only for OpenTURBO <<================
// ==>> TIDRV doesn't suppoty mutiple DBOPEN in TurboIMAGE Mode <<==
//
//*** DBXDBGIN Syntax = DBXBEGIN,BASE=id/BASELIST=id:id:id,Mode ***
//*** DBXEND   Syntax = DBXEND,BASE=id/BASELIST=id:id:id,Mode ***
//*** DBXUNDO  Syntax = DBXUNDO,BASE=id/BASELIST=id:id:id,Mode ***
//
DBXBEGIN,BASE=0,1
//
// Dataitem Level LOCK +++++ CC_PNLC_PO = X[36]
// ------------     EQUAL Confition    ------------------
DBLOCK,0,ARG='1,36,PURCHASE;,CC-PNLC-PO,=,&
CC_PNLC_PO_88                          ',5
DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_88,POR_KEY_88,88,88,88,&
88,88,88,REV_LOT,REQ_NO,COMMENT_IMS,88,USER_IMS,20020526,888888'
DBUNLOCK,0,,1
// ------------ <= and >= No ERROR when DBPUT ------------------
DBLOCK,0,ARG='2,36,PURCHASE;,CC-PNLC-PO,<=,&
CC_PNLC_PO_95                         ,36,PURCHASE;,CC-PNLC-PO,>=,&
CC_PNLC_PO_70                         ',5
DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_89,POR_KEY_89,89,89,89,&
89,89,89,REV_LOT,REQ_NO,COMMENT_IMS,89,USER_IMS,20020526,999999'
DBUNLOCK,0,,1
//
// Rewind Dataset PURCHASE
// ** Backward Serial DBGET 1 Records from PURCHASE and DBDELETE it
//
// Rewind Dataset PURCHASE and DBGET the Last 4 Records
//
DBCLOSE,0,PURCHASE,2
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
DBGET,0,PURCHASE;,3,LIST='@;',ARG=''
//
// DBXEND,BASE=0,1
DBXUNDO,BASE=0,1
//
DBCLOSE,0,,1
UNLOADTI
//
```

Output result file:

```
=>> Repeat[1] TICommand[LOADTI,ti1.ti]
=>> Repeat[1] TICommand[DBOPEN,INVENT.DATA.MOULTON;,FAVOR;,3]
DBOPEN,INVENT.DATA.MOULTON;,FAVOR;,3,status[1]=0,status[2]=1
```

```
=>> Repeat[1] TICommand[DBCONTROL,0,,7]
DBCONTROL: --------------------
mode=[7]
status[1]   =[0]
status[2]   =[1]
status[3-4] =[0]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBXBEGIN,BASE=0,1]
DBXBEGIN: --------------------
mode=[1]
status[1]   =[0]
status[2]   =[1]
status[3-4] =[0]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBLOCK,0,ARG='1,36,PURCHASE;,CC-PNLC-
PO,=,CC_PNLC_PO_88
                        ',5]
DBLOCK: --------------------
base=[INVENT.DATA.MOULTON;]
mode=[5]
status[1]   =[0]
status[2]   =[1]
status[3]   =[0]
status[4]   =[0]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1]
TICommand[DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_88,POR_KE
Y_88,88,88,88,88,88,88,REV_LOT,REQ_NO,COMMENT_IMS,88,USER_IMS,20020526,888
888']
DBPUT: --------------------
base=[INVENT.DATA.MOULTON;]
dset=PURCHASE;[15]
mode=[1]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_88
POR-KEY             1X20  =>POR_KEY_88
VEND-NO             1I2   =>+0000000088
QTY-ORD             1I2   =>+0000000088
DATE-ORD            1I2   =>+0000000088
DATE-PROM           1I2   =>+0000000088
DATE-LAST           1I2   =>+0000000088
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>COMMENT_IMS
UNIT-COST           1I4   =>+0000000000000000088
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000888888
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[500]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBUNLOCK,0,,1]
DBUNLOCK: --------------------
base=[INVENT.DATA.MOULTON;]
mode=[1]
status[1]   =[0]
status[2]   =[1]
status[3-4] =[500]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBLOCK,0,ARG='2,36,PURCHASE;,CC-PNLC-
PO,<=,CC_PNLC_PO_95
                       ,36,PURCHASE;,CC-PNLC-PO,>=,CC_PNLC_PO_70
       ',5]
DBLOCK: --------------------
base=[INVENT.DATA.MOULTON;]
mode=[5]
status[1]   =[0]
status[2]   =[1]
```

```
status[3]   =[0]
status[4]   =[500]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1]
TICommand[DBPUT,0,PURCHASE;,1,LIST='@;',DATA='CC_PNLC_PO_89,POR_KE
Y_89,89,89,89,89,89,89,REV_LOT,REQ_NO,COMMENT_IMS,89,USER_IMS,20020526,999
999']
DBPUT: --------------------
base=[INVENT.DATA.MOULTON;]
dset=PURCHASE;[15]
mode=[1]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_89
POR-KEY             1X20  =>POR_KEY_89
VEND-NO             1I2   =>+0000000089
(93/193) Continue?
QTY-ORD             1I2   =>+0000000089
DATE-ORD            1I2   =>+0000000089
DATE-PROM           1I2   =>+0000000089
QTY-RECD            1I2   =>+0000000089
DATE-LAST           1I2   =>+0000000089
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>COMMENT_IMS
UNIT-COST           1I4   =>+000000000000000089
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000999999
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[502]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBUNLOCK,0,,1]
DBUNLOCK: --------------------
base=[INVENT.DATA.MOULTON;]
mode=[1]
status[1]   =[0]
status[2]   =[0]
status[3-4] =[502]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBCLOSE,0,PURCHASE,2]
DBCLOSE,INVENT.DATA.MOULTON;[0],PURCHASE[15],2,db_status[1]=0
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
base=[INVENT.DATA.MOULTON;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_89
POR-KEY             1X20  =>POR_KEY_89
VEND-NO             1I2   =>+0000000089
QTY-ORD             1I2   =>+0000000089
DATE-ORD            1I2   =>+0000000089
DATE-PROM           1I2   =>+0000000089
QTY-RECD            1I2   =>+0000000089
DATE-LAST           1I2   =>+0000000089
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>COMMENT_IMS
UNIT-COST           1I4   =>+000000000000000089
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000999999
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[502]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBGET,0,PURCHASE;,3,LIST='@;',ARG='']
DBGET: --------------------
```

```
base=[INVENT.DATA.MOULTON;]
dset=PURCHASE;[15]
mode=[3]
list=@;
DATA BUFFER Begin ====================
CC-PNLC-PO          1X36  =>CC_PNLC_PO_88
POR-KEY             1X20  =>POR_KEY_88
VEND-NO             1I2   =>+0000000088
QTY-ORD             1I2   =>+0000000088
DATE-ORD            1I2   =>+0000000088
DATE-PROM           1I2   =>+0000000088
QTY-RECD            1I2   =>+0000000088
DATE-LAST           1I2   =>+0000000088
REV-LOT             1X10  =>REV_LOT
REQ-NO              1X16  =>REQ_NO
COMMENT             1X30  =>COMMENT_IMS
UNIT-COST           1I4   =>+000000000000000088
USER                1X8   =>USER_IMS
DATE-UPDT           1I2   =>+0020020526
TIME-UPDT           1I2   =>+0000888888
DATA BUFFER End   ====================
status[1]   =[0]
status[2]   =[80]
status[3-4] =[500]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBXUNDO,BASE=0,1]
DBXUNDO: --------------------
mode=[1]
status[1]   =[0]
status[2]   =[80]
status[3-4] =[500]
status[5-6] =[0]
status[7-8] =[0]
status[9-10]=[0]
=>> Repeat[1] TICommand[DBCLOSE,0,,1]
DBCLOSE,INVENT.DATA.MOULTON;[8224],[-1],1,db_status[1]=0
=>> Repeat[1] TICommand[UNLOADTI]
```

HP9000:

Run script:

```
export LTDBGOUT=-
$_OTB_BIN/tidrv –iintrx1 –oouttrx1
```

Input command file: Same as HP3000
Output result file: Same as HP3000

6. QUERY.PUB.SYS Program and CONFIG File: accesses ORACLE database remotely via OpenTURBO. HP e3000 only, unless HP ports QUERY over to HP-UX.

   The CONFIG file: you may use environment variable OT_CONFIG or file equation to redirect it, such as, ***SETVAR OT_CONFIG 'conf1'*** or ***FILE CONFIG = conf1***.

```
// This is used by QUERY.PUB.SYS - 2002-6-25 Lee Tsai
//
// For:  SYSTEM LEVEL CONFIGURATION FILE for OpenTURBO
//       This is used by HP/3000 Client SDK Version
// Date: 2002/04/18
//
// 515 is for HPUX and ORACLE
//
OT_TI_DBNAME = DEMO.DATA.IMAXSOFT
DEMO.DATA.IMAXSOFT {
DEMO.DATA {
DEMO {
   OT_ROOT_FILE         = DEMOti.ti.ims
   OT_RESERVE_WORD_FILE = RESERVE.ORACLE.ims
   OT_ERROR_FILE        = OTERROR.ORACLE.ims
   OT_HOST              = 207.92.64.8
   OT_SERVICE           = 32601
   OT_OS_RDBMS          = 515
   OT_RDB_LOGON         = demo_data_imaxsoft/diqzio
   OT_SDK_SERVER_PGM    = /users/lee/lbin/dbsvr
   OT_CIUPDATE          = ON
   OT_DUALMODE          = OFF
   TI_DUALMODE_HOST     = 207.92.64.66
   TI_DUALMODE_SERVICE  = 32608
   TI_DUALMODE_PRM      = DMDRV.BIN.IMAXSOFT
   OT_LOCKWAIT_CYCLE    = 60
   OT_TRX_THRESHOLD     = 10
OT_DETAILSETNAME = @ [
   OT_IGNORE_CHAINSTATUS  = OFF
   OT_BULKCHAINGET        = OFF
   OT_CHRONOLOGICAL       = ON
   OT_IGNORE_DBPUTSTATUS  = OFF
   OT_USE_IMAGERECNUM     = OFF
]
}
```

   QUERY program: you must first copy QUERY.PUB.SYS to QUERY.BIN.IMAXSOFT. Use QUERY.PUB.SYS to access TurboIMAGE database on HP e3000, use QUERY.BIN.IMAXSOFT to access ORACLE remotely via OpenTURBO emulator libraries.

   QUERY for ORACLE run script:

```
setvar ltdbgout '$stdlist'
file config=configq.lee.imaxsoft
run query.bin.imaxsoft;xl='otqry.a.imaxsoft,tidrv.a.imaxsoft,ltxl.pub.imaxsoft'
```

   1) Uses configq.lee.imaxsoft as the configuration file.
   2) Runs QUERY with OpenTURBO libraries, OTQRY.A.IMAXSOFT, TIDRV.A.IMAXSOFT and LTXL.PUB.IMAXSOFT.

   QUERY for ORACLE run script with debugging:

```
setvar ltdbg1 1
setvar ltdbg2 1
setvar ltdbg3 1
setvar ltdbg4 1
setvar ltdbg5 1
setvar ltdbg6 1
setvar ltdbg8 1
setvar ltdbg17 0
setvar ltdbg18 0
setvar ltdbg19 0
setvar ltdbgout '$stdlist'
file config=configq.lee.imaxsoft
run query.bin.imaxsoft;xl='otqrydbg.a.imaxsoft,tidrv.a.imaxsoft,ltxl.pub.imaxsoft'
```

```
1) Uses configq.lee.imaxsoft as the configuration file.
2) Runs QUERY with OpenTURBO libraries, OTQRYDBG.A.IMAXSOFT,
   TIDRV.A.IMAXSOFT and LTXL.PUB.IMAXSOFT.
```

7. otgenDB Program: converts the schema file created from HP3000 into ORACLE sqlplus format and adds security. All passwords must be entered in their encrypted form.

```
OpenTURBO otgenDB <A.01.03> iMaxsoft Corp. Copyright 2002

usage: otgenDB -d -u -k -p -s -f -x
       -u ORACLE Login User/Password
       -d TurboIMAGE Fully Qualified Database Name
          DBName.GROUP.ACCOUNT
       -k Input ORACLE Schema File from HP/3000
       -p ORACLE User DBName_GROUP_ACCOUNT Password
       -s Default Tablespace for storing New ORACLE User
          Security Objects that are replicated from
          TurboIMAGE Access Class and Password
       -x Output ORACLE Script File Name
       -f Use this option when this is the very first time
          to create the specified TurboIMAGE database into
          target ORACLE instance, which means the TurboIMAGE
          database never exist in the target ORACLE.
```

The run script file:

```
/users/lee/bin/otgenDB -uOT/JO \
                       -dMUSIC.TIDATA.IMAXSOFT \
                       -k../data/SMUSIC2.OTSCHEMA \
                       -pHPNDX \
                       -sMUSIC
```

The output script file:

OTBuildDB.EXECUTE:

```
sqlplus OT/OT @OTBuildDB.DROP
sqlplus OT/OT @/users/lee/musicdemo/data/SMUSIC2.OTSCHEMA
```

1) Drop all tables and MUSIC.TIDATA.IMS related objects from ORACLE database (OTBuildDB.DROP)

```
ALTER TABLE MUSIC_TIDATA_IMAXSOFT.ALBUMS         DROP PRIMARY KEY CASCADE;
DROP  TABLE MUSIC_TIDATA_IMAXSOFT.ALBUMS;
ALTER TABLE MUSIC_TIDATA_IMAXSOFT.COMPOSERS      DROP PRIMARY KEY CASCADE;
DROP  TABLE MUSIC_TIDATA_IMAXSOFT.COMPOSERS;
ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A   DROP PRIMARY KEY CASCADE;
DROP  TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A;
DROP  TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS;
DROP  TABLE MUSIC_TIDATA_IMAXSOFT.LOG;
COMMIT WORK;

DELETE FROM OT.TISET        WHERE TIBASE_NAME = 'MUSIC.TIDATA.IMAXSOFT';
DELETE FROM OT.TISET_KEY    WHERE TIBASE_NAME = 'MUSIC.TIDATA.IMAXSOFT';
DELETE FROM OT.TISET_FIELD  WHERE TIBASE_NAME = 'MUSIC.TIDATA.IMAXSOFT';
COMMIT WORK;
EXIT
```

2) Create all MUSIC.TIDATA.IMS tables and objects in ORACLE database (/users/lee/musicdemo/data/SMUSIC2.OTSCHEMA)

```
CREATE TABLE OT.TISET (
   TIBASE_NAME         VARCHAR2(26) NOT NULL,
   TABLE_NAME         VARCHAR2(256) NOT NULL,
   TISET_NAME          VARCHAR2(16) NOT NULL,
   TISET_NO              NUMBER(10) NOT NULL,
   TISET_TYPE          VARCHAR2(01) NOT NULL,
   LAST_SEQ_NO           NUMBER(20) NOT NULL,
   TIMESTAMP           VARCHAR2(14) NOT NULL
   );
```

```
CREATE TABLE OT.TISET_FIELD (
   TIBASE_NAME         VARCHAR2(26) NOT NULL,
   TISET_NO              NUMBER(10) NOT NULL,
   TIFIELD_SEQ_NO        NUMBER(10) NOT NULL,
   COLUMN_NAME        VARCHAR2(256) NOT NULL,
   TIITEM_NAME         VARCHAR2(20) NOT NULL,
   TIITEM_NO             NUMBER(10) NOT NULL,
   TIITEM_COUNT          NUMBER(10) NOT NULL,
   TIITEM_TYPE         VARCHAR2(01) NOT NULL,
   TIITEM_LENGTH         NUMBER(10) NOT NULL,
   COLUMN_TYPE           NUMBER(10) NOT NULL,
   COLUMN_LENGTH         NUMBER(10) NOT NULL,
   TIMESTAMP           VARCHAR2(14) NOT NULL
   );

CREATE TABLE OT.TISET_KEY (
   TIBASE_NAME         VARCHAR2(26) NOT NULL,
   TISET_NO              NUMBER(10) NOT NULL,
   TIKEY_SEQ_NO          NUMBER(10) NOT NULL,
   GEN_PATHCOLUMN      VARCHAR2( 1) NOT NULL,
   TIMASTERSET_NO        NUMBER(10) NOT NULL,
   TIINDEX_ITEM_NO       NUMBER(10) NOT NULL,
   COLUMN_NAME        VARCHAR2(256) NOT NULL,
   TIINDEX_ITEM_NAME   VARCHAR2(20) NOT NULL,
   TISORT_ITEM_NO        NUMBER(10) NOT NULL,
   TIMESTAMP           VARCHAR2(14) NOT NULL
   );

/************************************/
/* TIBASE:MUSIC.TIDATA.IMAXSOFT      */
/************************************/

/*************************/
/* TISET:ALBUMS           */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','ALBUMS','ALBUMS',1,'M',8,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',1,1,'N',1,1,'ALBUMCODE','ALBUMCODE',0,'2002071610
1418');

INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,1,'ALBUMCODE','ALBUMCODE',1,1,'I',2,1,10,'20020
716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,2,'ALBUMTITLE','ALBUMTITLE',3,1,'X',40,3,40,'20
020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,3,'MEDIUM','MEDIUM',12,1,'X',2,3,2,'20020716101
418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,4,'ALBUMCOST','ALBUMCOST',2,1,'P',8,1,7,'200207
16101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,5,'RECORDINGCO','RECORDINGCO',15,1,'X',16,3,16,
'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,6,'DATERECORDED','DATERECORDED',9,1,'X',16,3,16
,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,7,'MFGCODE','MFGCODE',13,1,'X',40,3,40,'2002071
6101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',1,8,'COMMENT_IMS','COMMENT',7,1,'X',80,3,80,'2002
0716101418');

/*************************/
/* TISET:COMPOSERS        */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','COMPOSERS','COMPOSERS',2,'M',4,'20020716101418')
;

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',2,1,'N',2,8,'COMPOSERNAME','COMPOSERNAME',0,'2002
0716101418');
```

```
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,1,'COMPOSERNAME','COMPOSERNAME',8,1,'X',16,3,16
,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,2,'BIRTH','BIRTH',5,1,'X',16,3,16,'200207161014
18');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,3,'DEATH','DEATH',10,1,'X',16,3,16,'20020716101
418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,4,'BIRTHPLACE','BIRTHPLACE',6,1,'X',40,3,40,'20
020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',2,5,'COMMENT_IMS','COMMENT',7,1,'X',80,3,80,'2002
0716101418');

/*************************/
/* TISET:SELECTIONS-A     */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','SELECTIONS_A','SELECTIONS-
A',3,'A',8,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',3,1,'N',3,16,'SELECTIONNAME','SELECTIONNAME',0,'2
0020716101418');

INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',3,1,'SELECTIONNAME','SELECTIONNAME',16,1,'X',40,3
,40,'20020716101418');

/*************************/
/* TISET:SELECTIONS       */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','SELECTIONS','SELECTIONS',4,'D',8,'20020716101418
');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',4,1,'Y',1,1,'ALBUMCODE','ALBUMCODE',0,'2002071610
1418');
INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',4,2,'Y',3,16,'SELECTIONNAME','SELECTIONNAME',0,'2
0020716101418');
INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',4,3,'Y',2,8,'COMPOSERNAME','COMPOSERNAME',0,'2002
0716101418');

INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,1,'ALBUMCODE','ALBUMCODE',1,1,'I',2,1,10,'20020
716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,2,'SELECTIONNAME','SELECTIONNAME',16,1,'X',40,3
,40,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,3,'COMPOSERNAME','COMPOSERNAME',8,1,'X',16,3,16
,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,4,'TIMING','TIMING',18,1,'X',16,3,16,'200207161
01418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,5,'PERFORMERS','PERFORMERS',14,1,'X',40,3,40,'2
0020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',4,6,'COMMENT_IMS','COMMENT',7,1,'X',80,3,80,'2002
0716101418');

/*************************/
/* TISET:LOG              */
/*************************/
INSERT INTO OT.TISET VALUES
('MUSIC.TIDATA.IMAXSOFT','LOG','LOG',5,'D',6,'20020716101418');

INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',5,1,'Y',1,1,'ALBUMCODE','ALBUMCODE',0,'2002071610
1418');
INSERT INTO OT.TISET_KEY VALUES
('MUSIC.TIDATA.IMAXSOFT',5,2,'Y',3,16,'SELECTIONNAME','SELECTIONNAME',0,'2
0020716101418');
```

```
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,1,'ALBUMCODE','ALBUMCODE',1,1,'I',2,1,10,'20020
716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,2,'SELECTIONNAME','SELECTIONNAME',16,1,'X',40,3
,40,'20020716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,3,'STARTTIME','STARTTIME',17,1,'X',16,3,16,'200
20716101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,4,'ENDTIME','ENDTIME',11,1,'X',16,3,16,'2002071
6101418');
INSERT INTO OT.TISET_FIELD VALUES
('MUSIC.TIDATA.IMAXSOFT',5,5,'ANNOUNCER','ANNOUNCER',4,1,'X',40,3,40,'2002
0716101418');

/***********************************/
/* TIBASE:MUSIC_TIDATA_IMAXSOFT      */
/***********************************/

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.ALBUMS (
   ALBUMCODE                      NUMBER(  10) NOT NULL, /* I[2] */
   ALBUMTITLE                  VARCHAR2(  40) NOT NULL, /* X[40] */
   MEDIUM                      VARCHAR2(   2) NOT NULL, /* X[2] */
   ALBUMCOST                      NUMBER(   7) NOT NULL, /* P[8] */
   RECORDINGCO                 VARCHAR2(  16) NOT NULL, /* X[16] */
   DATERECORDED                VARCHAR2(  16) NOT NULL, /* X[16] */
   MFGCODE                     VARCHAR2(  40) NOT NULL, /* X[40] */
   COMMENT_IMS                 VARCHAR2(  80) NOT NULL, /* X[80] */
   IMAXSOFT13_SEQ_NO              NUMBER(  20) NOT NULL
   );
   /* 8 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_ALBUMS ON
MUSIC_TIDATA_IMAXSOFT.ALBUMS (ALBUMCODE);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_ALBUMS ON
MUSIC_TIDATA_IMAXSOFT.ALBUMS (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.COMPOSERS (
   COMPOSERNAME                VARCHAR2(  16) NOT NULL, /* X[16] */
   BIRTH                       VARCHAR2(  16) NOT NULL, /* X[16] */
   DEATH                       VARCHAR2(  16) NOT NULL, /* X[16] */
   BIRTHPLACE                  VARCHAR2(  40) NOT NULL, /* X[40] */
   COMMENT_IMS                 VARCHAR2(  80) NOT NULL, /* X[80] */
   IMAXSOFT13_SEQ_NO              NUMBER(  20) NOT NULL
   );
   /* 4 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_COMPOSERS ON
MUSIC_TIDATA_IMAXSOFT.COMPOSERS (COMPOSERNAME);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_COMPOSERS ON
MUSIC_TIDATA_IMAXSOFT.COMPOSERS (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (
   SELECTIONNAME               VARCHAR2(  40) NOT NULL, /* X[40] */
   IMAXSOFT13_SEQ_NO              NUMBER(  20) NOT NULL
   );
   /* 8 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_SELECTIONS_A ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (SELECTIONNAME);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS_A ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS (
   ALBUMCODE                      NUMBER(  10) NOT NULL, /* I[2] */
   SELECTIONNAME               VARCHAR2(  40) NOT NULL, /* X[40] */
   COMPOSERNAME                VARCHAR2(  16) NOT NULL, /* X[16] */
   TIMING                      VARCHAR2(  16) NOT NULL, /* X[16] */
   PERFORMERS                  VARCHAR2(  40) NOT NULL, /* X[40] */
   COMMENT_IMS                 VARCHAR2(  80) NOT NULL, /* X[80] */
   IMAXSOFT13_PATH_01             NUMBER(  10) NOT NULL,
   IMAXSOFT13_PATH_02             NUMBER(  10) NOT NULL,
   IMAXSOFT13_PATH_03             NUMBER(  10) NOT NULL,
   IMAXSOFT13_SEQ_NO              NUMBER(  20) NOT NULL
   );
   /* 8 ROWS */
```

```
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (ALBUMCODE, IMAXSOFT13_PATH_01);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (SELECTIONNAME, IMAXSOFT13_PATH_02);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (COMPOSERNAME, IMAXSOFT13_PATH_03);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I04_SELECTIONS ON
MUSIC_TIDATA_IMAXSOFT.SELECTIONS (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.LOG (
   ALBUMCODE                    NUMBER(  10) NOT NULL, /* I[2] */
   SELECTIONNAME              VARCHAR2(  40) NOT NULL, /* X[40] */
   STARTTIME                  VARCHAR2(  16) NOT NULL, /* X[16] */
   ENDTIME                    VARCHAR2(  16) NOT NULL, /* X[16] */
   ANNOUNCER                  VARCHAR2(  40) NOT NULL, /* X[40] */
   IMAXSOFT13_PATH_01           NUMBER(  10) NOT NULL,
   IMAXSOFT13_PATH_02           NUMBER(  10) NOT NULL,
   IMAXSOFT13_SEQ_NO            NUMBER(  20) NOT NULL
   );
   /* 6 ROWS */

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_LOG ON
MUSIC_TIDATA_IMAXSOFT.LOG (ALBUMCODE, IMAXSOFT13_PATH_01);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_LOG ON
MUSIC_TIDATA_IMAXSOFT.LOG (SELECTIONNAME, IMAXSOFT13_PATH_02);

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_LOG ON
MUSIC_TIDATA_IMAXSOFT.LOG (IMAXSOFT13_SEQ_NO);

CREATE TABLE MUSIC_TIDATA_IMAXSOFT.DOOR_SYNC  (
   DBNAME           CHAR(30) NOT NULL,
   LOGRECNUM        INTEGER  NOT NULL,
   LOGFILENAME      CHAR(30) NOT NULL
   );

INSERT INTO MUSIC_TIDATA_IMAXSOFT.DOOR_SYNC VALUES ('-',0,' ');

COMMIT WORK;
EXIT;
```

8. otgenLOAD Program: converts table load script file created into ORACLE sqlplus format, adjusts steps for performance and adds security provisions. All passwords must be entered in their encrypted form.

```
OpenTURBO OTGenLoad <A.01.00> iMaxsoft Corp. Copyright 2002.

usage: otgenLOAD -u -d -t -c -x
       -u  ORACLE Login User/Password
       -d  TurboIMAGE Fully Qualified Database Name
           DBName.GROUP.ACCOUNT
       -t  A TABLE_NAME or @ for ALL_TABLES of the
           specified TurboIMAGE database
       -c  CACHE (default) or NOCACHE
           CACHE   is for ORACLE to get multiple sequence
                   numbers per request
           NOCACHE is for ORACLE to get single sequence
                   number at a time
           Use NOCACHE in all cases, unless there is a
           performance degradation
       -x  ORACLE Loader Script Output Filename
           Default = OTLoadDB.EXECUTE - executes all following
                                        scripts
                     OTLoadDB.DROP    - drops all table objects for
                                        performance
                     OTLoadDB.SQLLDR  - loads data into ORACLE
                                        database
                     OTLoadDB.GEN     - generates all table objects
                                        after successful loading
```

The run script file:

```
/users/lee/bin/otgenLOAD -u ot/JO                  \
                         -d MUSIC.TIDATA.IMAXSOFT \
                         -t @                      \
                         -c NOCACHE
```

The output script file:

OTLoadDB.EXECUTE:

```
sqlplus ot/ot @OTLoadDB.DROP
OTLoadDB.SQLLDR
sqlplus ot/ot @OTLoadDB.GEN
```

OTLoadDB.DROP: drops objects for speedy loading

```
ALTER TABLE MUSIC_TIDATA_IMAXSOFT.ALBUMS          ADD PRIMARY KEY (ALBUMCODE);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_ALBUMS
       ON MUSIC_TIDATA_IMAXSOFT.ALBUMS
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.COMPOSERS       ADD PRIMARY KEY (COMPOSERNAME);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_COMPOSERS
       ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A    ADD PRIMARY KEY
(SELECTIONNAME);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS_A
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (ALBUMCODE          , IMAXSOFT13_PATH_01);
```

```
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (SELECTIONNAME       , IMAXSOFT13_PATH_02);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (COMPOSERNAME        , IMAXSOFT13_PATH_03);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I04_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_LOG
       ON MUSIC_TIDATA_IMAXSOFT.LOG
       (ALBUMCODE           , IMAXSOFT13_PATH_01);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_LOG
       ON MUSIC_TIDATA_IMAXSOFT.LOG
       (SELECTIONNAME       , IMAXSOFT13_PATH_02);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_LOG
       ON MUSIC_TIDATA_IMAXSOFT.LOG
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS ADD (
   FOREIGN KEY(ALBUMCODE)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.ALBUMS (ALBUMCODE));
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS ADD (
   FOREIGN KEY(SELECTIONNAME)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (SELECTIONNAME));
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS ADD (
   FOREIGN KEY(COMPOSERNAME)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.COMPOSERS (COMPOSERNAME));
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.LOG ADD (
   FOREIGN KEY(ALBUMCODE)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.ALBUMS (ALBUMCODE));
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.LOG ADD (
   FOREIGN KEY(SELECTIONNAME)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (SELECTIONNAME));
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS
            INCREMENT BY 1 START WITH 8 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_ALBUMS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.ALBUMS
   FOR EACH ROW
BEGIN
   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS.NEXTVAL
             INTO :new.IMAXSOFT13_SEQ_NO FROM SYS.DUAL;
   END IF;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS
            INCREMENT BY 1 START WITH 4 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_COMPOSERS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS
   FOR EACH ROW
BEGIN
   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS.NEXTVAL
             INTO :new.IMAXSOFT13_SEQ_NO FROM SYS.DUAL;
   END IF;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A
            INCREMENT BY 1 START WITH 8 ORDER;
```

```
SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_SELECTIONS_A
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
   FOR EACH ROW
BEGIN
   SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A.NEXTVAL
     INTO :new.IMAXSOFT13_SEQ_NO FROM SYS.DUAL;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS
           INCREMENT BY 1 START WITH 8 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_SELECTIONS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
             INTO v_next_val   FROM SYS.DUAL;
      :new.IMAXSOFT13_SEQ_NO := v_next_val;
   END IF;
   IF (v_next_val = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
             INTO v_next_val   FROM SYS.DUAL;
   END IF;
   :new.IMAXSOFT13_PATH_01 := v_next_val;
   :new.IMAXSOFT13_PATH_02 := v_next_val;
   :new.IMAXSOFT13_PATH_03 := v_next_val;
END;
/
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TU1_SELECTIONS
   BEFORE UPDATE ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
   v_cnt      NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.ALBUMCODE <> :old.ALBUMCODE) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
                INTO v_next_val   FROM SYS.DUAL;
      END IF;
      :new.IMAXSOFT13_PATH_01 := v_next_val;
   END IF;

   IF (:new.SELECTIONNAME <> :old.SELECTIONNAME) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
                INTO v_next_val   FROM SYS.DUAL;
      END IF;
      IF (:new.IMAXSOFT13_PATH_02 = -888) THEN
         v_cnt := 0;
         SELECT COUNT(*)  INTO v_cnt
           FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
          WHERE SELECTIONNAME = :new.SELECTIONNAME;
         IF (v_cnt = 0) THEN
            INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
             VALUES(:new.SELECTIONNAME, v_cnt);
         END IF;
      END IF;
      :new.IMAXSOFT13_PATH_02 := v_next_val;
   END IF;

   IF (:new.COMPOSERNAME <> :old.COMPOSERNAME) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
                INTO v_next_val   FROM SYS.DUAL;
      END IF;
      :new.IMAXSOFT13_PATH_03 := v_next_val;
   END IF;
END;
```

```
/
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TA1_SELECTIONS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
   FOR EACH ROW
DECLARE
   v_cnt NUMBER;
BEGIN

   v_cnt := 0;
   SELECT COUNT(*)  INTO v_cnt
     FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
    WHERE SELECTIONNAME = :new.SELECTIONNAME;
   IF (v_cnt = 0) THEN
      INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
             VALUES(:new.SELECTIONNAME, v_cnt);
   END IF;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_LOG
           INCREMENT BY 1 START WITH 6 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_LOG
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.LOG
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
             INTO v_next_val   FROM SYS.DUAL;
      :new.IMAXSOFT13_SEQ_NO := v_next_val;
   END IF;
   IF (v_next_val = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
             INTO v_next_val   FROM SYS.DUAL;
   END IF;
   :new.IMAXSOFT13_PATH_01 := v_next_val;
   :new.IMAXSOFT13_PATH_02 := v_next_val;
END;
/
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TU1_LOG
   BEFORE UPDATE ON MUSIC_TIDATA_IMAXSOFT.LOG
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
   v_cnt      NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.ALBUMCODE <> :old.ALBUMCODE) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
                INTO v_next_val   FROM SYS.DUAL;
      END IF;
      :new.IMAXSOFT13_PATH_01 := v_next_val;
   END IF;

   IF (:new.SELECTIONNAME <> :old.SELECTIONNAME) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
                INTO v_next_val   FROM SYS.DUAL;
      END IF;
      IF (:new.IMAXSOFT13_PATH_02 = -888) THEN
         v_cnt := 0;
         SELECT COUNT(*)  INTO v_cnt
           FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
          WHERE SELECTIONNAME = :new.SELECTIONNAME;
         IF (v_cnt = 0) THEN
            INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
             VALUES(:new.SELECTIONNAME, v_cnt);
         END IF;
      END IF;
      :new.IMAXSOFT13_PATH_02 := v_next_val;
   END IF;
END;
/
```

```
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TA1_LOG
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.LOG
   FOR EACH ROW
DECLARE
   v_cnt NUMBER;
BEGIN

   v_cnt := 0;
   SELECT COUNT(*)  INTO v_cnt
     FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
    WHERE SELECTIONNAME = :new.SELECTIONNAME;
   IF (v_cnt = 0) THEN
      INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
             VALUES(:new.SELECTIONNAME, v_cnt);
   END IF;
END;
/
COMMIT WORK;
EXIT
```

OTLoadDB.SQLLDR: loads data into corresponding ORACLE table

```
sqlldr ot/ot rows=100 control=MUSIC01.CLT log=Log/MUSIC01.LOG;
sqlldr ot/ot rows=100 control=MUSIC02.CLT log=Log/MUSIC02.LOG;
sqlldr ot/ot rows=100 control=MUSIC03.CLT log=Log/MUSIC03.LOG;
sqlldr ot/ot rows=100 control=MUSIC04.CLT log=Log/MUSIC04.LOG;
sqlldr ot/ot rows=100 control=MUSIC05.CLT log=Log/MUSIC05.LOG;
```

OTLoadDB.GEN: re-creates all table objects after the loading

```
ALTER TABLE MUSIC_TIDATA_IMAXSOFT.ALBUMS           ADD PRIMARY KEY (ALBUMCODE);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_ALBUMS
       ON MUSIC_TIDATA_IMAXSOFT.ALBUMS
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.COMPOSERS        ADD PRIMARY KEY (COMPOSERNAME);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_COMPOSERS
       ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A     ADD PRIMARY KEY
(SELECTIONNAME);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS_A
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (ALBUMCODE            , IMAXSOFT13_PATH_01);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (SELECTIONNAME        , IMAXSOFT13_PATH_02);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (COMPOSERNAME         , IMAXSOFT13_PATH_03);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I04_SELECTIONS
       ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I01_LOG
       ON MUSIC_TIDATA_IMAXSOFT.LOG
       (ALBUMCODE            , IMAXSOFT13_PATH_01);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I02_LOG
       ON MUSIC_TIDATA_IMAXSOFT.LOG
       (SELECTIONNAME        , IMAXSOFT13_PATH_02);
CREATE UNIQUE INDEX MUSIC_TIDATA_IMAXSOFT.I03_LOG
       ON MUSIC_TIDATA_IMAXSOFT.LOG
       (IMAXSOFT13_SEQ_NO);
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS ADD (
   FOREIGN KEY(ALBUMCODE)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.ALBUMS (ALBUMCODE));
COMMIT WORK;
```

```
ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS ADD (
   FOREIGN KEY(SELECTIONNAME)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (SELECTIONNAME));
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.SELECTIONS ADD (
   FOREIGN KEY(COMPOSERNAME)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.COMPOSERS (COMPOSERNAME));
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.LOG ADD (
   FOREIGN KEY(ALBUMCODE)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.ALBUMS (ALBUMCODE));
COMMIT WORK;

ALTER TABLE MUSIC_TIDATA_IMAXSOFT.LOG ADD (
   FOREIGN KEY(SELECTIONNAME)
   REFERENCES MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A (SELECTIONNAME));
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS
            INCREMENT BY 1 START WITH 8 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_ALBUMS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.ALBUMS
   FOR EACH ROW
BEGIN
   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS.NEXTVAL
             INTO :new.IMAXSOFT13_SEQ_NO FROM SYS.DUAL;
   END IF;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS
           INCREMENT BY 1 START WITH 4 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_COMPOSERS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS
   FOR EACH ROW
BEGIN
   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS.NEXTVAL
             INTO :new.IMAXSOFT13_SEQ_NO FROM SYS.DUAL;
   END IF;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A
           INCREMENT BY 1 START WITH 8 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_SELECTIONS_A
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
   FOR EACH ROW
BEGIN
   SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A.NEXTVAL
     INTO :new.IMAXSOFT13_SEQ_NO FROM SYS.DUAL;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS
           INCREMENT BY 1 START WITH 8 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_SELECTIONS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
```

```
            INTO v_next_val   FROM SYS.DUAL;
      :new.IMAXSOFT13_SEQ_NO := v_next_val;
   END IF;
   IF (v_next_val = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
            INTO v_next_val   FROM SYS.DUAL;
   END IF;
   :new.IMAXSOFT13_PATH_01 := v_next_val;
   :new.IMAXSOFT13_PATH_02 := v_next_val;
   :new.IMAXSOFT13_PATH_03 := v_next_val;
END;
/
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TU1_SELECTIONS
   BEFORE UPDATE ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
   v_cnt      NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.ALBUMCODE <> :old.ALBUMCODE) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
               INTO v_next_val   FROM SYS.DUAL;
      END IF;
      :new.IMAXSOFT13_PATH_01 := v_next_val;
   END IF;

   IF (:new.SELECTIONNAME <> :old.SELECTIONNAME) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
               INTO v_next_val   FROM SYS.DUAL;
      END IF;
      IF (:new.IMAXSOFT13_PATH_02 = -888) THEN
         v_cnt := 0;
         SELECT COUNT(*)  INTO v_cnt
           FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
          WHERE SELECTIONNAME = :new.SELECTIONNAME;
         IF (v_cnt = 0) THEN
            INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
             VALUES(:new.SELECTIONNAME, v_cnt);
         END IF;
      END IF;
      :new.IMAXSOFT13_PATH_02 := v_next_val;
   END IF;

   IF (:new.COMPOSERNAME <> :old.COMPOSERNAME) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS.NEXTVAL
               INTO v_next_val   FROM SYS.DUAL;
      END IF;
      :new.IMAXSOFT13_PATH_03 := v_next_val;
   END IF;
END;
/
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TA1_SELECTIONS
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS
   FOR EACH ROW
DECLARE
   v_cnt NUMBER;
BEGIN

   v_cnt := 0;
   SELECT COUNT(*)  INTO v_cnt
     FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
    WHERE SELECTIONNAME = :new.SELECTIONNAME;
   IF (v_cnt = 0) THEN
      INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
            VALUES(:new.SELECTIONNAME, v_cnt);
   END IF;
END;
/
COMMIT WORK;

CREATE SEQUENCE MUSIC_TIDATA_IMAXSOFT.S01_LOG
           INCREMENT BY 1 START WITH 6 ORDER;
SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL FROM SYS.DUAL;

CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TI1_LOG
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.LOG
```

```
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.IMAXSOFT13_SEQ_NO = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
             INTO v_next_val   FROM SYS.DUAL;
      :new.IMAXSOFT13_SEQ_NO := v_next_val;
   END IF;
   IF (v_next_val = 0) THEN
      SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
             INTO v_next_val   FROM SYS.DUAL;
   END IF;
   :new.IMAXSOFT13_PATH_01 := v_next_val;
   :new.IMAXSOFT13_PATH_02 := v_next_val;
END;
/
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TU1_LOG
   BEFORE UPDATE ON MUSIC_TIDATA_IMAXSOFT.LOG
   FOR EACH ROW
DECLARE
   v_next_val NUMBER;
   v_cnt      NUMBER;
BEGIN
   v_next_val := 0;

   IF (:new.ALBUMCODE <> :old.ALBUMCODE) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
                INTO v_next_val   FROM SYS.DUAL;
      END IF;
      :new.IMAXSOFT13_PATH_01 := v_next_val;
   END IF;

   IF (:new.SELECTIONNAME <> :old.SELECTIONNAME) THEN
      IF (v_next_val = 0) THEN
         SELECT MUSIC_TIDATA_IMAXSOFT.S01_LOG.NEXTVAL
                INTO v_next_val   FROM SYS.DUAL;
      END IF;
      IF (:new.IMAXSOFT13_PATH_02 = -888) THEN
         v_cnt := 0;
         SELECT COUNT(*)  INTO v_cnt
           FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
          WHERE SELECTIONNAME = :new.SELECTIONNAME;
         IF (v_cnt = 0) THEN
            INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
             VALUES(:new.SELECTIONNAME, v_cnt);
         END IF;
      END IF;
      :new.IMAXSOFT13_PATH_02 := v_next_val;
   END IF;
END;
/
CREATE TRIGGER MUSIC_TIDATA_IMAXSOFT.TA1_LOG
   BEFORE INSERT ON MUSIC_TIDATA_IMAXSOFT.LOG
   FOR EACH ROW
DECLARE
   v_cnt NUMBER;
BEGIN

   v_cnt := 0;
   SELECT COUNT(*)  INTO v_cnt
     FROM MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
    WHERE SELECTIONNAME = :new.SELECTIONNAME;
   IF (v_cnt = 0) THEN
      INSERT INTO MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A
             VALUES(:new.SELECTIONNAME, v_cnt);
   END IF;
END;
/
COMMIT WORK;

EXIT
```

9. tigenSEC Program: reads OpenTURBO TI root-file and replicates TurboIMAGE security, which includes database passwords, data set and data item access classes, to ORACLE environment. All passwords must be entered in their encrypted form.

   You must put RESERVE.ORACLE file in directory $_OTB_ROOT/conf or use -r to identify the location of file RESERVE.ORACLE.

```
   OpenTURBO tigenSEC <A.01.00> iMaxsoft Corp. Copyright 2002.
   usage: tigenSEC -d -t -r -p -s -x
      -d     TurboIMAGE Fully Qualified Database Name
             DBName.GROUP.ACCOUNT.
      -t     OpenTURBO TI Rootfile
      -r     ORACLE RESERVE WORD File
      -p     ORACLE User DBName_GROUP_ACCOUNT Password
      -s     Default Tablespace for storing ORACLE User
             Security Objects that are replicated from
             TurboIMAGE Access Class and Password
      -x     Output ORACLE Security Script File Name

      *NOTE* You must have OpenTURBO TI Rootfile in the
             local directory or a link to the TI Roofile.
             In OpenTURBO, the fully qualified TurboIMAGE
             database name is the TI Rootfile name. So, either
             you name your TI Roofile as the fully qualified
             TurboIMAGE name, or make a link to it.
             For example:
                 ln -s ../db/tiDBName.ti DBNAME.GROUP.ACCOUNT
```

   The run script file:

```
   /users/lee/bin/tigenSEC -d MUSIC.TIDATA.IMAXSOFT \
                           -t MUSIC.TIDATA.IMAXSOFT \
                           -r /users/lee/conf/RESERVE.ORACLE  \
                           -p HPNDX \
                           -s MUSIC
```

   The output script file:

   OTCreateSEC.EXECUTE:

```
      sqlplus MUSIC_TIDATA_IMAXSOFT/MUSIC @OTCreateSEC.sql
```

   OTCreateSEC.sql:

```
DROP   TABLE MUSIC_TIDATA_IMAXSOFT.DBACCESS;
CREATE TABLE MUSIC_TIDATA_IMAXSOFT.DBACCESS (
       PID       INTEGER       NOT NULL,
       DBMODE    INTEGER       NOT NULL,
       MEMO      VARCHAR2(35)  NOT NULL,
       TIMESTAMP VARCHAR2(14)  NOT NULL);

DROP   TABLE MUSIC_TIDATA_IMAXSOFT.DBLOCK;
CREATE TABLE MUSIC_TIDATA_IMAXSOFT.DBLOCK (
       PID       INTEGER       NOT NULL,
       SEQ       INTEGER       NOT NULL,
       MEMO      VARCHAR2(35)  NOT NULL,
       TIMESTAMP VARCHAR2(14)  NOT NULL);

DROP   TABLE MUSIC_TIDATA_IMAXSOFT.DBLOCK1;
CREATE TABLE MUSIC_TIDATA_IMAXSOFT.DBLOCK1 (
       PID       INTEGER       NOT NULL,
       SEQ       INTEGER       NOT NULL,
       LKLEVEL   INTEGER       NOT NULL,
       DBASE     INTEGER       NOT NULL,
       DSET      INTEGER       NOT NULL,
       ITEM      INTEGER       NOT NULL,
       OP        INTEGER       NOT NULL);
```

```
DROP   TABLE MUSIC_TIDATA_IMAXSOFT.DBLOCK2;
CREATE TABLE MUSIC_TIDATA_IMAXSOFT.DBLOCK2 (
       PID       INTEGER       NOT NULL,
       SEQ       INTEGER       NOT NULL,
       DBASE     INTEGER       NOT NULL,
       DSET      INTEGER       NOT NULL,
       ITEM      INTEGER       NOT NULL,
       OP        INTEGER       NOT NULL,
       TYPE      INTEGER       NOT NULL,
       VLEN      INTEGER       NOT NULL,
       VAL       VARCHAR2(256) NOT NULL);

DROP   USER MUSIC_TIDATA_IMAXSOFT_10 CASCADE;
CREATE USER MUSIC_TIDATA_IMAXSOFT_10 IDENTIFIED BY MGR DEFAULT TABLESPACE
MUSIC;
DROP   USER MUSIC_TIDATA_IMAXSOFT_20 CASCADE;
CREATE USER MUSIC_TIDATA_IMAXSOFT_20 IDENTIFIED BY DIR DEFAULT TABLESPACE
MUSIC;
DROP   USER MUSIC_TIDATA_IMAXSOFT_30 CASCADE;
CREATE USER MUSIC_TIDATA_IMAXSOFT_30 IDENTIFIED BY ANNCR DEFAULT TABLESPACE
MUSIC;
COMMIT WORK;

GRANT CREATE SESSION to MUSIC_TIDATA_IMAXSOFT_10;
GRANT CREATE SYNONYM to MUSIC_TIDATA_IMAXSOFT_10;
GRANT CREATE SESSION to MUSIC_TIDATA_IMAXSOFT_20;
GRANT CREATE SYNONYM to MUSIC_TIDATA_IMAXSOFT_20;
GRANT CREATE SESSION to MUSIC_TIDATA_IMAXSOFT_30;
GRANT CREATE SYNONYM to MUSIC_TIDATA_IMAXSOFT_30;
COMMIT WORK;

GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_30;
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.ALBUMS         to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_30;
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.COMPOSERS      to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A   to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A   to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS_A   to
MUSIC_TIDATA_IMAXSOFT_30;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_30;
```

```
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.SELECTIONS     to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_30;
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_10;
GRANT INSERT ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_30;
GRANT UPDATE ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_30;
GRANT DELETE ON MUSIC_TIDATA_IMAXSOFT.LOG            to
MUSIC_TIDATA_IMAXSOFT_30;
COMMIT WORK;

CONNECT MUSIC_TIDATA_IMAXSOFT_10/MGR;
CREATE SYNONYM DBACCESS FOR MUSIC_TIDATA_IMAXSOFT.DBACCESS;
CREATE SYNONYM DBLOCK   FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK;
CREATE SYNONYM DBLOCK1  FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK1;
CREATE SYNONYM DBLOCK2  FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK2;
COMMIT WORK;

CONNECT MUSIC_TIDATA_IMAXSOFT_20/DIR;
CREATE SYNONYM DBACCESS FOR MUSIC_TIDATA_IMAXSOFT.DBACCESS;
CREATE SYNONYM DBLOCK   FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK;
CREATE SYNONYM DBLOCK1  FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK1;
CREATE SYNONYM DBLOCK2  FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK2;
COMMIT WORK;

CONNECT MUSIC_TIDATA_IMAXSOFT_30/ANNCR;
CREATE SYNONYM DBACCESS FOR MUSIC_TIDATA_IMAXSOFT.DBACCESS;
CREATE SYNONYM DBLOCK   FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK;
CREATE SYNONYM DBLOCK1  FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK1;
CREATE SYNONYM DBLOCK2  FOR MUSIC_TIDATA_IMAXSOFT.DBLOCK2;
COMMIT WORK;

CONNECT MUSIC_TIDATA_IMAXSOFT/MUSIC;
GRANT ALL ON DBACCESS TO MUSIC_TIDATA_IMAXSOFT_10;
GRANT ALL ON DBLOCK   TO MUSIC_TIDATA_IMAXSOFT_10;
GRANT ALL ON DBLOCK1  TO MUSIC_TIDATA_IMAXSOFT_10;
GRANT ALL ON DBLOCK2  TO MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS       TO
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS    TO
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A TO
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS   TO
MUSIC_TIDATA_IMAXSOFT_10;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_LOG          TO
MUSIC_TIDATA_IMAXSOFT_10;
GRANT ALL ON DBACCESS TO MUSIC_TIDATA_IMAXSOFT_20;
GRANT ALL ON DBLOCK   TO MUSIC_TIDATA_IMAXSOFT_20;
GRANT ALL ON DBLOCK1  TO MUSIC_TIDATA_IMAXSOFT_20;
GRANT ALL ON DBLOCK2  TO MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS       TO
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS    TO
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A TO
MUSIC_TIDATA_IMAXSOFT_20;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS   TO
MUSIC_TIDATA_IMAXSOFT_20;
```

```
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_LOG          TO
MUSIC_TIDATA_IMAXSOFT_20;
GRANT ALL ON DBACCESS TO MUSIC_TIDATA_IMAXSOFT_30;
GRANT ALL ON DBLOCK   TO MUSIC_TIDATA_IMAXSOFT_30;
GRANT ALL ON DBLOCK1  TO MUSIC_TIDATA_IMAXSOFT_30;
GRANT ALL ON DBLOCK2  TO MUSIC_TIDATA_IMAXSOFT_30;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_ALBUMS       TO
MUSIC_TIDATA_IMAXSOFT_30;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_COMPOSERS    TO
MUSIC_TIDATA_IMAXSOFT_30;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS_A TO
MUSIC_TIDATA_IMAXSOFT_30;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_SELECTIONS   TO
MUSIC_TIDATA_IMAXSOFT_30;
GRANT SELECT ON MUSIC_TIDATA_IMAXSOFT.S01_LOG          TO
MUSIC_TIDATA_IMAXSOFT_30;
COMMIT WORK;
EXIT
```

10 otANALYZER Program: produces a series of reports to validate the integrity of your ORACLE database, to analyze your ORACLE database for performance enhancements, and to check the accuracy of OpenTURBO ORACLE migration. All passwords must be entered in their encrypted form.

```
OpenTURBO otANALYZER <A.01.00> iMAXSOFT Corp. Copyright 2002

usage:otALANYZER -u -d -r -o
      -u ORACLE Login User/Password
      -d TurboIMAGE Fully Qualified Database Name
         DBName.GROUP.ACCOUNT
      -r Report ID, you may specify one or many separated
         by comma; such as -r1 or -r1,2,3
         Report ID Report Description
         ========= ========================================
             1     Detail Dataset Chain Length Analysis
             2     ORACLE Tablespaces Allocation Analysis
             3     Post-Migration Dataset Count Validation
      -o Report Output Filename, Default = Report
```

The run script file:

```
 /users/lee/bin/otANALYZER -d MUSIC.TIDATA.IMAXSOFT \
                           -u ot/JO \
                           -r 1,2,3 \
                           -o /users/lee/musicdemo/report/DBReport
```

The output script file:

Report 1 – Detail Dataset Path Analysis:

```
Report  : PATH ANALYSIS
ImageDB : MUSIC.TIDATA.IMAXSOFT                          Date: July 14 2002
Owner   : MUSIC_TIDATA_IMAXSOFT

                                                  Path Length Statistic
    Table                  Index Column           Min   Max/BOF  Avg/Total
---------------------- ----------------------- -------- -------- ----------
ACCOUNT_FILE           ACCOUNT                        1        1       1.00
                                                ---------------- ----------
                                                               1        955*
                                                ---------------- ----------
ACCOUNT_FILE2          ACCOUNT                        1        1       1.00
                                                ---------------- ----------
                                                               1        415*
                                                ---------------- ----------
ACCOUNT_FILE3          ACCOUNT                        1        1       1.00
                                                ---------------- ----------
                                                               1        239*
                                                ---------------- ----------
SEARCH_KEY_FILE        SEARCH_KEY                     1        1       1.00
                                                ---------------- ----------
                                                               1        761*
. . .
```

Report 2 – ORACLE Tablespace Requirement Analysis:

```
Report  : TABLESPACE REQUIREMENT
ImageDB : MUSIC.TIDATA.IMAXSOFT                      Date: July 14 2002
Owner   : MUSIC_TIDATA_IMAXSOFT

    Table              Tablespace(DATA) kbytes  Tablespace(INDEX) kbytes
---------------------- ------------------------ -------------------------
ACCOUNT_FILE                                12                         10
ACCOUNT_FILE2                                5                          4
ACCOUNT_FILE3                                3                          2
SEARCH_KEY_FILE                             15                          8
SSN_FILE                                    14                         10
AUTO_KEY_X8                                  0                          0
MEMBER_FILE                                960                         30
ACC_FILE                                     4                          1
```

NEXT_ACC_FILE	0	0
SHARE_FILE	1502	71
IRA_FILE	94	9
LOAN_FILE	1300	31
COLLATERAL_FILE	17	2
LN_PUR_FILE	2	1

. . .

Report 3 – OpenTURBO Load Analysis:

Report : SEQUENCE NO
ImageDB : MUSIC.TIDATA.IMAXSOFT Date: July 14 2002
Owner : MUSIC_TIDATA_IMAXSOFT

Table	TYPE	IMAXSOFT13_SEQ_NO	SEQUENCE INITIAL #
ACCOUNT_FILE	A	955	955
ACCOUNT_FILE2	A	415	415
ACCOUNT_FILE3	A	239	239
SEARCH_KEY_FILE	A	761	761
SSN_FILE	A	971	971
AUTO_KEY_X8	A	6	6
MEMBER_FILE	D	913	913
ACC_FILE	M	35	35
NEXT_ACC_FILE	M	0	0
SHARE_FILE	D	1900	1900
IRA_FILE	D	282	282
LOAN_FILE	D	1103	1103
COLLATERAL_FILE	M	75	75

. . .

11 otCLEAN Program: deletes Automatic table entries that are not foreign to any Detail tables. OpenTURBO delete function doesn't delete keys without details automatically, so you must run this program periodically to clean-up any dangling keys from Automatic tables. The reason is performance.

TurboIMAGE Automatic dataset has absolutely no value in the ORALCE environment, OpenTURBO maps them into ORALCE indexes which serve the sole purpose of fast key search; unless you have application requires serial scan of those Automatic datasets, otherwise, do not migrate Automatic datasets.

All passwords must be entered in their encrypted form.

```
OpenTURBO otCLEAN <A.01.03> iMaxsoft Corp. Copyright 2002

usage: otCLEAN -u -d -r -o
       -u ORACLE OpenTURBO Superuser Login OT/Password
       -d TurboIMAGE Fully Qualified Database Name
          DBName.GROUP.ACCOUNT - to be cleaned
       -r 1 = Clean Up Automatic Tables Only
          2 = Reserved
          3 = Reserved
       -o Execution Log File Name, Default = otCLEAN.log
```

The run script file:

```
/users/lee/lbin/otCLEAN -d MUSIC.TIDATA.IMAXSOFT \
                        -u ot/JO \
                        -r 1 \
                        -o /users/lee/musicdemo/log/MUSICClean.log
```

```
OpenTURBO otCLEAN <A.01.00> iMAXSOFT Corp. Copyright 2002

ORACLE_USER  - [ot/JO]
TIBASE_NAME  - [MEMBRS.DMDATA.IMS] [MEMBRS_DMDATA_IMS]
OUTPUT_FILE  - [/users/lee/demo/log/MEMBRSClean.log]


   SET NAME               RECORD COUNT
 ---------------------- --------------
 ACCOUNT_FILE                      955
 ACCOUNT_FILE2                     415
 ACCOUNT_FILE3                     239
 SEARCH_KEY_FILE                   761
 SSN_FILE                          971
 AUTO_KEY_X8                         6
 FLAG_LEVEL_FILE                    69
 PACKAGE_FILE                       13
 M_VIN_NUMBER                       81
 M_VIN_SUB_NBR                      60

 OK to process Y/[N] ? >> y


 ACCOUNT_FILE     (     955) (2002-07-17 15:39)     1/10
              ...       955  (2002-07-17 15:40)

 ACCOUNT_FILE2    (     415) (2002-07-17 15:40)     2/10
(DELETE FROM MEMBRS_DMDATA_IMS.ACCOUNT_FILE2 WHERE IMAXSOFT13_SEQ_NO = 10) - 0
              ...       415  (2002-07-17 15:40)

 ACCOUNT_FILE3    (     239) (2002-07-17 15:40)     3/10
              ...       239  (2002-07-17 15:40)

 SEARCH_KEY_FILE  (     761) (2002-07-17 15:40)     4/10
              ...       761  (2002-07-17 15:41)

 SSN_FILE         (     971) (2002-07-17 15:41)     5/10
              ...       971  (2002-07-17 15:41)
```

```
AUTO_KEY_X8      (       6) (2002-07-17 15:41)     6/10
              ...        6  (2002-07-17 15:41)

FLAG_LEVEL_FILE  (      69) (2002-07-17 15:41)     7/10
              ...       69  (2002-07-17 15:41)

PACKAGE_FILE     (      13) (2002-07-17 15:41)     8/10
(DELETE FROM MEMBRS_DMDATA_IMS.PACKAGE_FILE WHERE IMAXSOFT13_SEQ_NO = 1) - 0
(DELETE FROM MEMBRS_DMDATA_IMS.PACKAGE_FILE WHERE IMAXSOFT13_SEQ_NO = 2) - 0
(DELETE FROM MEMBRS_DMDATA_IMS.PACKAGE_FILE WHERE IMAXSOFT13_SEQ_NO = 3) - 0
(DELETE FROM MEMBRS_DMDATA_IMS.PACKAGE_FILE WHERE IMAXSOFT13_SEQ_NO = 5) - 0
(DELETE FROM MEMBRS_DMDATA_IMS.PACKAGE_FILE WHERE IMAXSOFT13_SEQ_NO = 6) - 0
              ...       13  (2002-07-17 15:41)

M_VIN_NUMBER     (      81) (2002-07-17 15:41)     9/10
              ...       81  (2002-07-17 15:41)

M_VIN_SUB_NBR    (      60) (2002-07-17 15:41)    10/10
              ...       60  (2002-07-17 15:41)
```

The log file:

```
*** Clean_up Automatic Set Record ***
TIBASE_NAME  - [MEMBRS_DMDATA_IMS]

ACCOUNT_FILE     (     955) (2002-07-17 15:39)     1/10
              ...      955  (2002-07-17 15:40)

ACCOUNT_FILE2    (     415) (2002-07-17 15:40)     2/10
              ...      415  (2002-07-17 15:40)

ACCOUNT_FILE3    (     239) (2002-07-17 15:40)     3/10
              ...      239  (2002-07-17 15:40)

SEARCH_KEY_FILE  (     761) (2002-07-17 15:40)     4/10
              ...      761  (2002-07-17 15:41)

SSN_FILE         (     971) (2002-07-17 15:41)     5/10
              ...      971  (2002-07-17 15:41)

AUTO_KEY_X8      (       6) (2002-07-17 15:41)     6/10
              ...        6  (2002-07-17 15:41)

FLAG_LEVEL_FILE  (      69) (2002-07-17 15:41)     7/10
              ...       69  (2002-07-17 15:41)

PACKAGE_FILE     (      13) (2002-07-17 15:41)     8/10
              ...       13  (2002-07-17 15:41)

M_VIN_NUMBER     (      81) (2002-07-17 15:41)     9/10
              ...       81  (2002-07-17 15:41)

M_VIN_SUB_NBR    (      60) (2002-07-17 15:41)    10/10
              ...       60  (2002-07-17 15:41)
```

13 OpenTURBO emulator emulates almost 100% of TurboIMAGE calls, for details please refer to HP/TurboIMAGE/XL Database management System Reference Manual (Part Number: 30391-99011). Any discrepancies or specific to OpenTURBO emulator are addressed in this chapter.

OpenTURBO vs. TurboIMAGE General Differences:

DBBEGIN	Not Supported
DBCLOSE	100%
DBCONTROL	Supported Mode: 5 Turn CIUPDATE ON 6 Turn CIUPDATE OFF 7 Allow Dynamic Multiple DB Transaction OpenTURBO Mode: 88 Set OpenTURBO Debugger Level Remotely 89 Set OpenTURBO Debugger Output Filename Remotely 90 Set IGNORE_RETURN_STATUS ON 91 Set IGNORE_RETURN_STATUS OFF 92 Set ONE_DIRECTION_CHAIN_GET ON 93 Set ONE_DIRECTION_CHAIN_GET OFF 94 Enlarge MAX_CURSOR_ALLOWED
DBDELETE	100%
DBEND	Not Supported
DBERROR	100%
DBEXPLAIN	100%
DBFIND	100% - Mode 10 (Not certified by TPI vendors)
DBGET	100%
DBINFO	100% - Except the dataset spaces allocation related info which is not applicable in ORACLE
DBLOCK	100%
DBMEMO	Not Supported
DBOPEN	100%
DBPUT	100% - Except the path info is not updated, OpenTURBO doesn't not automatically adjust the current path linkage after the addition for continuing Mode 5 and 6 DBGET
DBUNLOCK	100%
DBUPDATE	100% - CIUPDATE is supported
DBXBEGIN	100%
DBXEND	100%
DBXUNDO	100%

OpenTURBO Performance Enhancements:

DBBEGIN	Commit Work and initiate a OpenTURBO transaction
DBCLOSE	N/A
DBCONTROL	N/A
DBDELETE	Elimination of AUTOMATIC Dataset
DBEND	Commit Work the OpenTURBO transaction initiated by DBBEGIN
DBERROR	In-memory Client or Server message mapping capability
DBEXPLAIN	In-memory Client or Server message mapping capability
DBFIND	Ignore status words 3 through 10, this option triggers subsequent DGBET(5/6) in NO-CURSOR, BULK-FETCH and PRE-FETCH mode.
DBGET	Bulk fetch option supports one direction chain get DBGET(5 or 6) after a DBFIND, which is either forward chain get or backward chain get, but not both forward and backward in the same chain. This option triggers OpenTURBO BULK-FETCH and PRE-FETCH mode. Bulk fetch option also improves DBGET(2/3) serial get performance significantly, it triggers not only OpenTURBO BULK-FTECH and PRE-FETCH feature, it also triggers ORACLE pre-fetch between its client and server communication, with proper memory allocation, the performance can be improved by over 10 times.
DBINFO	N/A
DBLOCK	Thick Client LOCK Manager Thin Server LOCK Manager
DBMEMO	Not Supported
DBOPEN	One DBSVR per DBOPEN The password - you must terminate it with a ';' character, when its length is less 8 bytes long.
DBPUT	Elimination of AUTOMATIC Dataset
DBUNLOCK	N/A
DBUPDATE	Elimination of AUTOMATIC Dataset No performance degradation for CIUPDATE, same as NON-CIUPDATE
DBXBEGIN	N/A
DBXEND	N/A
DBXUNDO	N/A

OpenTURBO Additional Features:

DBBEGIN	Commit Work and initiate a OpenTURBO transaction

DBCLOSE	N/A
DBCONTROL	88 - Set OpenTURBO Debugger Level Remotely qualifier[0] = Debugger Level, valid values are 1 through 31 qualifier[1] = 1 (ON), 0 (OFF) qualifier[0] and qualifier[1] are 2-bytes short 89 - Set OpenTURBO Debugger Output Filename Remotely qualifier[] contains Debugger Output Filename 90 - Set IGNORE_RETURN_STATUS ON qualifier[] contains Dataset Name or Dataset Number (2-bytes short); if the first byte contains an @, sets all Datasets 91 - Set IGNORE_RETURN_STATUS OFF qualifier[] contains Dataset Name or Dataset Number (2-bytes short); if the first byte contains an @, sets all Datasets 92 - Set ONE_DIRECTION_CHAIN_GET ON qualifier[] contains Dataset Name or Dataset Number (2-bytes short); if the first byte contains an @, sets all Datasets 93 - Set ONE_DIRECTION_CHAIN_GET OFF qualifier[] contains Dataset Name or Dataset Number (2-bytes short); if the first byte contains an @, sets all Datasets 94 - Enlarge MAX_CURSOR_ALLOWED qualifier[0] = Max Number of Concurrent Cursors (2-bytes short)
DBDELETE	N/A
DBEND	Commit Work the OpenTURBO transaction initiated by DBBEGIN
DBERROR	SQL Error
DBEXPLAIN	SQL Explain
DBFIND	Exact, wildcard, range, relational operators search are all supported by default, no need for special settings
DBGET	
DBINFO	406 - Return Buffer's Half-word 16, 17, 18 and 19 are used for OpenTURBO Version, such as 'A.01.04 ' 406 - Return Buffer's Half-word 25 and 26 (starts from 1) contains the Number Of Concurrent DBOPEN including the calling process 8001 – OpenTURBO Client Library Version 8002 – OpenTURBO Server Program Version (DBSVR) 8003 – RDBMS and OS Type, 515 for ORACLE on HP-UX 8004 – RDBMS Database Name 8005 – Host Name where RDBMS Database resides 8006 – Service or Port Number that is used by OpenTURBO Listener 8007 – RDBMS Logon 8008 – OpenTURBO Server Program Full Name 8009 – OpenTURBO TI Root-File Name 8010 – OpenTURBO TI Root-File Version 8011 – OpenTURBO CONFIG File Name 8012 – OpenTURBO Reserve Word File Name 8013 – OpenTURBO Error Message File Name 8014 – Unconditional LOCK Pause and Re-try Count 8015 – CIUPDATE Allow Indicator 8016 – DBXBEGIN Allow Indicator 8017 – DUALMODE Indicator 8018 – DUALMODE HP/3000 Host Name 8019 – DUALMODE HP/3000 Listener Service or Port Number 8020 – DUALMODE HP/3000 Server Program Name (DMDRV) 8021 - IMAGEMODE Indicator - Access TurboIMAGE Only 8022 - IGNORE_RETURN_STATUS Indicator (IGNORE_CHAINSTATUS) Qualifier contains a Dataset Name or Dataset Number Return Buffer 1st Half-word contains TRUE=1 or FALSE=0 8023 - ONE_DIRECTION_CHAIN_GET Indicator (BULKCHAINGET) Qualifier contains a Dataset Name or Dataset Number Return Buffer 1st Half-word contains TRUE=1 or FALSE=0 8081 – Total SQL Calls since DBOPEN 8082 – Total Concurrent DBOPENs in the Client Process 8083 – Total Concurrent Open Cursors for the Database 8084 – Number of Chunks for the OpenTURBO Internal Cursor Pool Chunk Size is 30000 Bytes
DBLOCK	Entity LOCK support for ORACLE applications (ORACLE LOCK Object) OpenTURBO and ORACLE applications co-existence support
DBMEMO	Not Supported
DBOPEN	OpenTURBO and ORACLE applications co-existence support
DBPUT	N/A
DBUNLOCK	N/A
DBUPDATE	Update all fields (columns) always
DBXBEGIN	2PC Support A.02.00
DBXEND	2PC Support A.02.00
DBXUNDO	2PC Support A.02.00

14 OpenTURBO Listener: you must starts this daemon process on you HP9000 regardless you access ORACLE database locally or remotely. The listener program accepts DBOPEN requests from your application programs, then spawns the OpenTURBO server process DBSVR which performs all subsequent database access calls. The listener is also responsible for OpenTURBO recovery, if DBSVR aborts abnormally, the listener will make sure all dangling database objects that are created by the DBSVR are clean-up properly.

You must provide an unused server port for listener to use, check /etc/services file and find an open number, the range is from 1 through 32768. It is highly recommend that you add the newly assigned entry into /etc/services file for ease of control.

Here is an example entry:

```
OTB   32608/tcp   otb     # For OpenTURBO Listener
```

In the client, the listener connection control data is stored in the CONFIG file, OT_HOST, OT_SERVICE, OT_OS_RDBMS, OT_RDB_LOGON and OT_SDK_SERVER_PRG are used to connect to the target host machine, to talk to the listener, to spawn the server program, and to connect to the proper database via proper database logon.

Note, the OT_RDB_LOGON is used only when your program login as the creator of the TurboIMAGE and use semicolon as the password, otherwise, the DBOPEN password is mapped to the its corresponding ORACLE user.

Sample script to start a listener:

```
export LTDBG17=0
export LTDBG18=0
export LTDBG19=0
export LTDBG27=0
export LTDBG28=0
export LTDBGOUT=-
$_OTB_BIN/listner 32601
```

You need to start the listener with super user capability and from the login with proper ORACLE and OpenTURBO setup, which means all environment variables, access paths, library paths, etc. must point to the proper places for ORACLE database and OpenTURBO, MS-COBOL, ORACLE dynamic libraries.

Do not turn on OpenTURBO debugging from listener level, turn it on through DBCONTROL; if you turn on OpenTURBO debugging at listener level, the LTDBGOUT file logs all clients info, there is no way to isolate individual client trace. This feature is used only in the development environment, you can assign each programmer a listener, then he or she controls their own environment, one client per listener, this is an easy way to turn on OpenTURBO trace.

15 OpenTURBO Lanutil: server process management tool. It allows you to view any process that is currently connected to your ORACLE database locally or remotely through OpenTURBO. This utility can be run on the net, as long as you specify the host name and service port number, it directs you all the way to the specific listener and reports the status.

Lanutil is also used to gracefully shut down the listener, command STOPALL.

Example:

```
LANUTIL (A.06.00.00) iMAXSOFT Corp. Copyright 1993-2002, All
Rights Reserved.


HOST:[127.0.0.1]  APPLICATION:[32601]

Commands: LIST    - shows all connected users.
          KILL id - kills the specified user.
          STOPALL - terminates listener and all users.
          HOST id - sets to new host node name.
          APPL id - sets to new application name.
          SETQ qname #servers
                  - sets # of standby servers for a queue
          EXIT    - ends the LANUTIL.
LANUTIL>>
```

1) LIST command, shows all server processes that are spawned by the listener running on the HOST [127.0.0.1] and waiting on the SERVICE [32601].
2) KILL id command, kills the specific server process via the ID from the LIST command report.
3) STOPALL, is the best way to gracefully shut down this listener along with associated child processes.
4) HOST id command, reconnects Lanutil to another HOST via either an IP address or a DNS name.
5) APPL id command, reconnects Lanutil to another listener via either a SERVICE number or a SERVICE name.
6) SETQ qname command, OpenTURBO listener supports persistent and parallel stand-by modes, at current release of OpenTURBO, only persistent listener is supported, ignore this command.
7) EXIT command, ends Lanutil program.

OpenTURBO otDBUTIL: subset of TurboIMAGE DBUTIL tool. It lists DBOPEN processes and DBLOCK requests from your ORACLE database to the same TurboIMAGE logical database. This utility can be run on the net and uses the same CONFIG file as other OpenTURBO libraries and utilities, and the CONFIG can be redirected via file equation or environment variable OT_CONFIG.

Example:

```
:FILE CONFIG=CONFIG.LEE.IMAXSOFT
:otDBUTIL

OpenTURBO DBUTIL <A.01.02> iMaxsoft Corp. Copyright 2002.

>>help

EXIT                      : Leave the program
SHOW (database-name) USERS: Display all current OpenTURBO users
SHOW (database-name) LOCKS: Display all current OpenTURBO locks

>>show invent users

For database INVENT.DATA.MOULTON

PIN   PATH      EXECUTING PROGRAM      JOBNUM MODE
===== ==== ========================== ====== ====
 2227  1           OpenTURBO DBOPEN          1
 2254  1           OpenTURBO DBOPEN          1

>>show invent locks

For database INVENT.DATA.MOULTON

LOCKED ENTITY                                    PIN       PROGRAM
============================================== ===== ================
DATA SET    ITMMAST . . . . . . . . . . .        2227 OpenTURBO DBLOCK
    ITMDTL:  CC-PARTNO  =   MOXIMG14047         2254  OpenTURBO DBLOCK

>>
```

16 OpenTURBO otSETChecksum: sets the checksum into ORACLE database and ensures that your ORACLE database structure is in sync with OpenTURBO TIFile. You must run this program to sync ORACLE and TIFIle, whenever there is changes to TIFile.

```
OpenTURBO otSETChecksum <A.01.00> iMaxsoft Corp. Copyright 2002

usage: otSETChecksum -u -d -t -v
       -u ORACLE Login User/Password
       -d TurboIMAGE Fully Qualified Database Name
          DBName.GROUP.ACCOUNT.
       -t TIFile Name
       -v View Checksum Summary

Example to set checksum:

   otSETChecksum -u OT/JO            \
                 -d MUSIC.DATA.TICHG \
                 -t MUSIC.DATA.TICHG

Example to view checksum:

   otCHECKChecksum -u OT/JO            \
                   -d MUSIC.DATA.TICHG \
                   -t MUSIC.DATA.TICHG \
                   -v

   OpenTURBO otSETChecksum <A.01.00> iMaxsoft Corp. Copyright 2002

   ORACLE_USER  - [OT/JO]
   TIBASE_NAME  - [MUSIC.DATA.TICHG]
   TIFile       - [MUSIC.DATA.TICHG]
   View         - [1]

    ====================== ========================== ==========================
                           TIBASE                     TIFile
    ---------------------- -------------------------- --------------------------
    TIBASE_NAME            MUSIC.DATA.TICHG           MUSIC.DATA.TICHG
    TIFILE_NAME            MUSIC.DATA.TICHG           MUSIC.DATA.TICHG
    TIFILE_VERSION         A.01.03                    A.01.03
    CHECKSUM_ITEMNO        253                        253
    CHECKSUM_SETNO         518                        518
    CHECKSUM_PATHNO        204                        204
    TIMESTAMP              20020910105145             20020910105145
```

```
OpenTURBO otCHECKChecksum: verifies and compares checksums in ORACLE
database and TIFile.

OpenTURBO otCHECKChecksum <A.01.03> iMaxsoft Corp. Copyright 2002

usage: otCHECKChecksum -u -d -t
       -u ORACLE Login User/Password
       -d TurboIMAGE Fully Qualified Database Name
          DBName.GROUP.ACCOUNT.
       -t TIFile Name

Example to set checksum:

   /users/lee/bin/otCHECKChecksum -u OT/JO \
                                  -d MUSIC.TIDATA.IMAXSOFT \
                                  -t ../tidb/ti

   OpenTURBO otCHECKChecksum <A.01.03> iMaxsoft Corp. Copyright 2002

   oracle_user  - [OT/JO]
   tibase_name  - [MUSIC.TIDATA.IMAXSOFT]
   tifile       - [../tidb/ti]

   Checksum Summary Report:

   ---------- ------- FROM DATABASE -------- --------- FROM TIFile --------
   DBNAME    : MUSIC.TIDATA.IMAXSOFT          MUSIC.TIDATA.IMAXSOFT
   TIFILE    : ../tidb/ti                     ../tidb/ti
   VERSION   : A.01.04                        A.01.04
   ITEM-CKSUM: 171                            171
   DSET-CKSUM: 305                            305
   PATH-CKSUM: 96                             96

   otCHECKChecksum Result = 1
                            1 = Mactched
                            0 = Not Mactched
                           -1 = Error
```

```
17 OpenTURBO otDBPURGE: purge a logical TurboIMAGE database from an
   ORACLE instance. You must purge the old database before you can reuse
   the same database name, a fully qualified TurboIMAGE database name.
   If -x is used, two script files will be generated,
   OTPurgeDB_test.EXECUTE and OTPurgeDB_test.SQL, you must manually
   start OTPurgeDB_text.EXECUTE to purge the database. If -x is omitted,
   the database will be purged in real-time. All passwords must be
   entered in their encrypted form.

   OpenTURBO otDBPURGE <A.01.03> iMaxsoft Corp. Copyright 2002

   usage: otDBPURGE -u -t -p -d -x
          -u ORACLE Login User/Password
          -d TurboIMAGE Fully Qualified Database Name
             DBName.GROUP.ACCOUNT.
          -t TIFile Name
          -p New ORACLE User DBName_GROUP_ACCOUNT Password
          -x Output ORACLE Script File Name

   Example:

      otDBPURGE -u OT/JO            \
                -d MUSIC.COPY.TICHG \
                -t /users/lee/otb/tdrv/MUSIC.COPY.TICHG \
                -p HPNDX            \
                -x test
```

```
OpenTURBO otDBCOPY: make a copy of a logical TurboIMAGE database in
ORACLE environment. If -x is used, three script files will be
generated, OTCopyDB_test.EXECUTE, OTCopyDB_test.SQL and
OTCopyDB_test.CLASS, you must manually start OTCopyDB_text.EXECUTE to
copy a database. If -x is omitted, the database will be copied in
real-time. All passwords must be entered in their encrypted form.

OpenTURBO otDBCOPY <A.01.00> iMaxsoft Corp. Copyright 2002

usage: otDBCOPY -u -d -t -r -n -o -p -s -x
       -u ORACLE Login User/Password
       -d TurboIMAGE Fully Qualified Database Name
          DBName.GROUP.ACCOUNT.
       -t Old TIFile Name
       -r ORACLE RESERVE WORD File Name
       -n New DataBase Name
       -o New TIFile Name
       -p New ORACLE User DBName_GROUP_ACCOUNT Password
       -s Default Tablespace for storing New ORACLE User
          Security Objects that are replicated from
          TurboIMAGE Access Class and Password
       -x Output ORACLE Script File Name

Example:

   #export LTDBG1=1
   #export LTDBG2=1
   #export LTDBGOUT=-
   otDBCOPY -u OT/JO            \
            -d MUSIC.DATA.TICHG \
            -t MUSIC.DATA.TICHG \
            -r /users/lee/conf/RESERVE.ORACLE \
            -n MUSIC.COPY.TICHG \
            -o NEWTI            \
            -p HPNDX            \
            -s MUSIC_TICHG      \
            -x test

   1) otDBCOPY generates a set of scripts file for database copy. The
      -x option value will be used as a part for the script file
      name, it is not the script file name.
   2) Out script files are OTCopyDB_test.CLASS, OTCopyDB_test.SQL,
      and OTCopyDB_test.EXECUTE.
   3) You only need to execute script OTCopyDB_test.EXECUTE.
```

18 OpenTURBO otINTEGRITY program:

otINTEGRITY scans and checks ORACLE database objects created by OpenTURBO at TurboIMAGE level to ensure the integrity of your ORACLE database and TIFile. otINTEGRITY reports any integrity related discrepancies and errors, you must shutdown the database and fix all errors to prevent any further damages to your ORACLE database. All passwords must be entered in their encrypted form.

OpenTURBO otINTEGRITY <A.01.04> iMaxsoft Corp. Copyright 2002

```
usage: otINTEGRITY -u -d -t -r -p
       -u ORACLE Login User/Encrypted Password
       -d TurboIMAGE Fully Qualified Database Name
          DBName.GROUP.ACCOUNT.
       -t TI File Name
       -r ORACLE RESERVE WORD File Name
       -p ORACLE Encrypted Password.for User
          DBName_GROUP_ACCOUNT.
```

Database Integrities:

Database structure:

a. Table: table name, table type (logical type, A, M, and D), and table columns which includes column name, column position, column type, column length and NULL indicator.

b. Index: index name, index type (unique, non-unique, cluster, etc.) and key columns which includes column name and column position.

c. Constraints: primary keys and foreign keys, which is mainly used for maintaining A and M to D relationships.

Database unique number:

d. Sequence: compare the maximum value of all OpenTURBO internal control columns IMAXSOFT13_PATH_nn and IMAXSOFT13_SEQ_NO to the next_val of table sequence object and to ensure there is no unique constrain violation.

Database triggers:

e. Trigger: trigger name, trigger type (insert, update or delete), and trigger contents.

Database securities:

f. Authority: access authorities for OpenTURBO control tables DBACCESS, DBLOCK, DBLOCK1, DBLOCK2, and DOOR_SYNC. ORACLE users and passwords (TurboIMAGE classes and passwords), and ORACLE user's database objects access authorities, objects are tables, columns, sequences, synonym (name domain), session, DBA, and DDL.

Database checksums:

g. otCHECKChecksum - verifies checksums in ORACLE OT.TIBASE and OpenTURBO TIFile.

otINTEGRITY views - you must create OpenTURBO system views first:

```
$cd /users/lee/admin/view
```

```
$sqlplus
. . .
. . . <login in ot/ot>
. . .
SQL> @all_view.sql;
SQL> commit work;
SQL> exit;
```

Script - all_view.sql:

```
@own_tbl
@tbl_col
@tbl_con
@tbl_con_col
@tbl_idx
@tbl_idx_col
@tbl_role
@tbl_role_dba
@tbl_sec
@tbl_sec_col
@tbl_seq
@tbl_synonyms
@tbl_system_priv
@tbl_trigger
```

Script - drop_view.sql:

```
drop view OT.OT_TABLE;
drop view OT.OT_COLUMN;
drop view OT.OT_CONSTRAINT;
drop view OT.OT_CON_COLUMN;
drop view OT.OT_INDEX;
drop view OT.OT_IND_COLUMN;
drop view OT.OT_ROLE;
drop view OT.OT_ROLE_DBA;
drop view OT.OT_TBL_SECURITY;
drop view OT.OT_COL_SECURITY;
drop view OT.OT_SEQUENCE;
drop view OT.OT_SYNONYM;
drop view OT.OT_SYS_PRIVILEGE;
drop view OT.OT_TRIGGER;
```

Script - own_tbl.sql:

```
rem View Name  :OT.OT_TABLE
rem Definition :To Get All The OpenTurbo Table Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
rem Certified  :2002-11-06 Lee Tsai
rem
create or replace view OT.OT_TABLE as
select u.name AS OWNER, o.name AS TABLE_NAME
 from  sys.user$ u, sys.obj$ o, sys.tab$ t
where  o.owner# = u.user#
  and  o.obj#   = t.obj#
/
```

Script - tbl_col.sql:

```
rem View Name  :OT.OT_COLUMN
rem Definition :To Get All The OpenTurbo Column Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_COLUMN as
select u.name AS OWNER_NAME, o.name AS TABLE_NAME, c.name AS COLUMN_NAME,
       decode(c.type#, 1, decode(c.charsetform, 2, 'NVARCHAR2', 'VARCHAR2'),
                       2, decode(c.scale, null,
                                 decode(c.precision#, null, 'NUMBER', 'FLOAT')
                                 , 'NUMBER'),
                       8, 'LONG', 9, decode(c.charsetform,
                                            2, 'NCHAR VARYING', 'VARCHAR'),
                       12, 'DATE', 23, 'RAW', 24, 'LONG RAW',
```

```
                          69, 'ROWID',
                          96, decode(c.charsetform, 2, 'NCHAR', 'CHAR'),
                          105, 'MLSLABEL',
                          106, 'MLSLABEL',
                          111, o.name,
                          112, decode(c.charsetform, 2, 'NCLOB', 'CLOB'),
                          113, 'BLOB', 114, 'BFILE', 115, 'CFILE',
                          121, o.name,
                          122, o.name,
                          123, o.name,
                          178, 'TIME(' ||c.scale|| ')',
                          179, 'TIME(' ||c.scale|| ')' || ' WITH TIME ZONE',
                          180, 'TIME(' ||c.scale|| ')',
                          181, 'TIME(' ||c.scale|| ')' || ' WITH TIME ZONE',
                          231, 'TIME(' ||c.scale|| ')' || ' WITH TIME ZONE',
                          182, 'INTERVAL YEAR(' ||c.precision#|| ') TO MONTH',
                          183, 'INTERVAL DAY(' ||c.precision#|| ') TO SECOND(' ||
                               c.scale || ')',
                          208, 'UROWID',
                          'UNDEFINED')  AS DATA_TYPE,
          c.length AS DATA_LENGTH, c.precision# AS DATA_PRECISION,
          c.scale  AS DATA_SCALE,
          decode(sign(c.null$), -1, 'D', 0, 'NULL', 'NOT NULL') AS ISNULL,
          c.col# AS COLUMN_ID
    from sys.col$ c, sys.user$ u, sys.obj$ o
    where o.owner# = u.user#
      and o.obj#   = c.obj#
/
```

Script - tbl_con.sql:

```
rem View Name  :OT.OT_CONSTRAINT
rem Definition :To Get All The OpenTurbo Constraint Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_CONSTRAINT as
select ou.name as OWNER          ,oc.name as CONSTRAINT_NAME,
       decode(c.type#, 1, 'C', 2, 'P', 3, 'U',
              4, 'R', 5, 'V', 6, 'O', 7 , 'C', '?') as C_TYPE,
       o.name  as TABLE_NAME        ,ru.name  as R_OWNER,
       rc.name as R_CONSTRAINT_NAME
  from sys.con$ oc, sys.con$ rc, sys.user$ ou, sys.user$ ru,
       sys.obj$ o,  sys.cdef$ c
 where oc.owner# = ou.user#
   and oc.con#   = c.con#
   and c.obj#    = o.obj#
   and c.type#  != 8
   and c.rcon#   = rc.con#(+)
   and rc.owner# = ru.user#(+)
/
```

Script - tbl_con_col.sql:

```
rem View Name  :OT.OT_CON_COLUMN
rem Definition :To Get All The OpenTurbo Constrain Column Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_CON_COLUMN as
select u.name as OWNER           ,c.name CONSTRAINT_NAME,
       o.name as TABLE_NAME,
       decode(bitand(col.property, 524289), 1, ac.name, 524288, ac.name,
              col.name) as COLUMN_NAME, cc.pos# as POSITION
  from sys.user$ u, sys.con$ c, sys.col$ col, sys.ccol$ cc, sys.cdef$ cd,
       sys.obj$ o,  sys.attrcol$ ac
 where c.owner#    = u.user#
   and c.con#      = cd.con#
   and cd.con#     = cc.con#
   and cc.obj#     = col.obj#
   and cc.intcol#  = col.intcol#
   and cc.obj#     = o.obj#
   and col.obj#    = ac.obj#(+)
   and col.intcol# = ac.intcol#(+)
/
```

Script - tbl_idx.sql:

```
rem View Name  :OT.OT_INDEX
rem Definition :To Get All The OpenTurbo INDEX Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_INDEX as
select u.name  as OWNER       , o.name  as INDEX_NAME,
       iu.name as TABLE_OWNER , io.name as TABLE_NAME,
       decode(bitand(i.property, 16), 0, '', 'FUNCTION-BASED ') ||
        decode(i.type#, 1, 'NORMAL' ||
                           decode(bitand(i.property, 4), 0, '', 4, '/REV'),
                       2, 'BITMAP', 3, 'CLUSTER', 4, 'IOT - TOP',
                       5, 'IOT - NESTED', 6, 'SECONDARY', 7, 'ANSI', 8,
'LOB',
                       9, 'DOMAIN') as INDEX_TYPE
  from sys.user$ u, sys.obj$ o, sys.user$ iu, sys.obj$ io, sys.ind$ i
 where o.owner#  = u.user#
   and o.obj#    = i.obj#
   and i.bo#     = io.obj#
   and io.owner# = iu.user#
/
```

Script - tbl_idx_col.sql:

```
rem View Name  :OT.OT_IND_COLUMN
rem Definition :To Get All The OpenTurbo Index Column Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_IND_COLUMN as
select io.name as OWNER       , idx.name  as INDEX_NAME,
       decode(bitand(i.property, 16), 0, '', 'FUNCTION-BASED ') ||
        decode(i.type#, 1, 'NORMAL' ||
                           decode(bitand(i.property, 4), 0, '', 4, '/REV'),
                       2, 'BITMAP', 3, 'CLUSTER', 4, 'IOT - TOP',
                       5, 'IOT - NESTED', 6, 'SECONDARY', 7, 'ANSI',
                       8, 'LOB',
                       9, 'DOMAIN') INDEX_TYPE,
       bo.name as TABLE_OWNER , base.name as TABLE_NAME,
       decode(bitand(c.property, 524289), 1, ac.name, 524288, ac.name,
                     c.name) as COLUMN_NAME,
       decode(bitand(c.property, 131072), 131072, 'DESC', 'ASC') as COL_ORDER
  from sys.col$ c,   sys.obj$ idx, sys.obj$ base, sys.icol$ ic,
       sys.user$ io, sys.user$ bo, sys.ind$ i,    sys.attrcol$ ac
 where ic.bo#     = c.obj#
   and ic.intcol# = c.intcol#
   and ic.bo#     = base.obj#
   and io.user#   = idx.owner#
   and bo.user#   = base.owner#
   and ic.obj#    = idx.obj#
   and idx.obj#   = i.obj#
   and i.type# in (1, 2, 3, 4, 6, 7, 9)
   and c.obj#     = ac.obj#(+)
   and c.intcol#  = ac.intcol#(+)
/
```

Script - tbl_role.sql:

```
rem View Name  :OT.OT_ROLE
rem Definition :To Get All The OpenTurbo Role Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_ROLE as
select decode(sa.grantee#, 1, 'PUBLIC', u1.name) as GRANTEE,
       u2.name  as GRANTED_ROLE
  from sys.sysauth$ sa, sys.user$ u1, sys.user$ u2, sys.defrole$ ud
  where sa.grantee#     =ud.user#(+)
    and sa.privilege#   =ud.role#(+) and u1.user#=sa.grantee#
    and u2.user#        =sa.privilege#
  group by decode(sa.grantee#,1,'PUBLIC',u1.name), u2.name
/
```

Script - tbl_role_dba.sql:

```
rem View Name  :OT.OT_ROLE_DBA
rem Definition :To Get All The DBA Role Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_ROLE_DBA as
select decode(sa.grantee#, 1, 'PUBLIC', u1.name) as GRANTEE,
       u2.name as GRANTED_ROLE
  from sys.sysauth$ sa, sys.user$ u1, sys.user$ u2, sys.defrole$ ud
  where sa.grantee#     =ud.user#(+)
    and sa.privilege#   =ud.role#(+) and u1.user#=sa.grantee#
    and u2.user#        =sa.privilege#
    and u2.name         ='DBA'
  group by decode(sa.grantee#,1,'PUBLIC',u1.name), u2.name
/
```

Script - tbl_sec.sql:

```
rem View Name  :OT.OT_TBL_SECURITY
rem Definition :To Get All The OpenTurbo Table Granted Security
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view ot.OT_TBL_SECURITY as
select ue.name  as GRANTEE    , u.name  as OWNER,
       o.name   as TABLE_NAME , ur.name as GRANTOR,
       tpm.name as PRIVILEGE
  from sys.objauth$ oa, sys.obj$ o, sys.user$ u, sys.user$ ur,
sys.user$ ue,
       table_privilege_map tpm
 where oa.obj#       = o.obj#
   and oa.grantor#   = ur.user#
   and oa.grantee#   = ue.user#
   and oa.col# is null
   and oa.privilege# = tpm.privilege
   and u.user#       = o.owner#
/
```

Script - tbl_sec_col.sql:

```
rem View Name  :OT.OT_COL_SECURITY
rem Definition :To Get All The OpenTurbo Column Granted Security
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_COL_SECURITY as
select ue.name  as GRANTEE    , u.name  as OWNER,
       o.name   as TABLE_NAME , c.name  as COLUMN_NAME,
       ur.name  as GRANTOR    ,
       tpm.name as PRIVILEGE
  from sys.objauth$ oa, sys.obj$ o, sys.user$ u, sys.user$ ur,  sys.user$ ue,
       sys.col$ c,      table_privilege_map tpm
 where oa.obj#       = o.obj#
   and oa.grantor#   = ur.user#
   and oa.grantee#   = ue.user#
   and oa.obj#       = c.obj#
   and oa.col#       = c.col#
   and bitand(c.property, 32) = 0
   and oa.col# is null
   and oa.privilege# = tpm.privilege
   and u.user#       = o.owner#
/
```

Script - tbl_seq.sql:

```
rem View Name  :OT.OT_SEQUENCE
rem Definition :To Get All The OpenTurbo Sequence Names
rem
rem Author     :Steven Tsung
```

```
rem Date       :10/23/2002
rem
rem Certified  :2002-11-06 Lee Tsai
rem
create or replace view OT.OT_SEQUENCE as
select u.name       as OWNER  , o.name      as SEQUENCE_NAME,
       s.minvalue   as MIN    , s.maxvalue  as MAX,
       s.increment$ as INC    ,
       decode(s.cycle#, 0, 'N', 1, 'Y')     as CYCLE,
       decode(s.order$, 0, 'N', 1, 'Y')     as ORDER_NO,
       s.cache      as ISCACHE, s.highwater as LAST_NUMBER
  from sys.seq$ s, sys.obj$ o, sys.user$ u
  where u.user#  = o.owner#
    and o.obj#   = s.obj#
/
```

Script - tbl_synonyms.sql:

```
rem View Name  :OT.OT_SYNONYM
rem Definition :To Get All The OpenTurbo Synonym Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_SYNONYM as
select u.name  as OWNER      , o.name  as SYNONYM_NAME,
       s.owner as TABLE_OWNER, s.name  as TABLE_NAME
  from sys.user$ u, sys.syn$ s, sys.obj$ o
 where o.obj#   = s.obj#
   and o.type#  = 5
   and o.owner# = u.user#
/
```

Script - tbl_system_priv.sql:

```
rem View Name  :OT.OT_SYS_PRIVILEGE
rem Definition :To Get All The OpenTurbo System Privilege
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view ot.ot_sys_privilege as
select u.name as GRANTEE  , spm.name as PRIVILEGE
  from sys.system_privilege_map spm, sys.sysauth$ sa, sys.user$ u
 where sa.grantee#   = u.user#
   and sa.privilege# = spm.privilege
/
```

Script - tbl_trigger.sql:

```
rem View Name  :OT.OT_TRIGGER
rem Definition :To Get All The OpenTurbo Trigger Names
rem
rem Author     :Steven Tsung
rem Date       :10/23/2002
rem
create or replace view OT.OT_TRIGGER as
select trigusr.name as OWNER      , trigobj.name as TRIGGER_NAME,
       tabusr.name  as TABLE_OWNER, tabobj.name  as TABLE_NAME,
       decode(t.insert$*100 + t.update$*10 + t.delete$,
              100, 'INSERT',
              010, 'UPDATE',
              001, 'DELETE',
              110, 'INSERT OR UPDATE',
              101, 'INSERT OR DELETE',
              011, 'UPDATE OR DELETE',
              111, 'INSERT OR UPDATE OR DELETE', 'ERROR') as TRIGGER_EVENT,
       t.definition as TRIGGER_CONTENT
  from sys.obj$  trigobj, sys.obj$  tabobj, sys.trigger$ t,
       sys.user$ tabusr,  sys.user$ trigusr
 where trigobj.obj#   = t.obj#
   and tabobj.obj#    = t.baseobject
   and tabobj.owner#  = tabusr.user#
   and trigobj.owner# = trigusr.user#
/
```

Example:

```
$_OTB_BIN/otINTEGRITY -u OT/JO \
                      -d MUSIC.TIDATA.IMAXSOFT \
                      -r /users/lee/conf/RESERVE.ORACLE \
                      -t /users/lee/musicdemo/tidb/ti \
                      -p HPNDX > output_music
```

otINTEGRITY Summary Report:

```
OpenTURBO otINTEGRITY <A.01.04> iMaxsoft Corp. Copyright 2002

ORACLE_USER      - [OT/JO]
TIBASE_NAME      - [MUSIC.TIDATA.IMAXSOFT] [MUSIC_TIDATA_IMAXSOFT]
TIFile           - [/users/lee/musicdemo/tidb/ti]
RWFile           - [/users/lee/conf/RESERVE.ORACLE]
OWNER_PASSWD     - [HPNDX]

----- ORACLE TABLES of TurboIMAGE[MUSIC.TIDATA.IMAXSOFT]:
DBIntegrity - [ALBUMS                  ] OT_SETS_NUM=[0]
DBIntegrity - [COMPOSERS               ] OT_SETS_NUM=[1]
DBIntegrity - [DBACCESS                ] OT_SETS_NUM=[2]
DBIntegrity - [DBLOCK                  ] OT_SETS_NUM=[3]
DBIntegrity - [DBLOCK1                 ] OT_SETS_NUM=[4]
DBIntegrity - [DBLOCK2                 ] OT_SETS_NUM=[5]
DBIntegrity - [DOOR_SYNC               ] OT_SETS_NUM=[6]
DBIntegrity - [LOG                     ] OT_SETS_NUM=[7]
DBIntegrity - [SELECTIONS              ] OT_SETS_NUM=[8]
DBIntegrity - [SELECTIONS_A            ] OT_SETS_NUM=[9]

----- ORACLE TABLE Count = [10], TurboIMAGE Set Count = [5]

***** TurboIMAGE Dataset=[ALBUMS][M ][1] Table=[ALBUMS]:
OT_GetSetInfo - [ALBUMCODE             ] COLUMNSCnt=[0]
OT_GetSetInfo - [ALBUMTITLE            ] COLUMNSCnt=[1]
OT_GetSetInfo - [MEDIUM                ] COLUMNSCnt=[2]
OT_GetSetInfo - [ALBUMCOST             ] COLUMNSCnt=[3]
OT_GetSetInfo - [RECORDINGCO           ] COLUMNSCnt=[4]
OT_GetSetInfo - [DATERECORDED          ] COLUMNSCnt=[5]
OT_GetSetInfo - [MFGCODE               ] COLUMNSCnt=[6]
OT_GetSetInfo - [COMMENT_IMS           ] COLUMNSCnt=[7]
OT_GetSetInfo - [IMAXSOFT13_SEQ_NO     ] COLUMNSCnt=[8]

===== TABLE[ALBUMS] Column Summary:
WARN [OT_GetSetInfo] Column[IMAXSOFT13_SEQ_NO     ] has no name match.

===== SEQUENCE=S01_ALBUMS CurrValue=18

***** TurboIMAGE Dataset=[COMPOSERS][M ][2] Table=[COMPOSERS]:
OT_GetSetInfo - [COMPOSERNAME          ] COLUMNSCnt=[0]
OT_GetSetInfo - [BIRTH                 ] COLUMNSCnt=[1]
OT_GetSetInfo - [DEATH                 ] COLUMNSCnt=[2]
OT_GetSetInfo - [BIRTHPLACE            ] COLUMNSCnt=[3]
OT_GetSetInfo - [COMMENT_IMS           ] COLUMNSCnt=[4]
OT_GetSetInfo - [IMAXSOFT13_SEQ_NO     ] COLUMNSCnt=[5]

===== TABLE[COMPOSERS] Column Summary:
WARN [OT_GetSetInfo] Column[IMAXSOFT13_SEQ_NO     ] has no name match.

===== SEQUENCE=S01_COMPOSERS CurrValue=11

***** TurboIMAGE Dataset=[SELECTIONS-A][A ][3] Table=[SELECTIONS_A]:
OT_GetSetInfo - [SELECTIONNAME         ] COLUMNSCnt=[0]
OT_GetSetInfo - [IMAXSOFT13_SEQ_NO     ] COLUMNSCnt=[1]

===== TABLE[SELECTIONS_A] Column Summary:
WARN [OT_GetSetInfo] Column[IMAXSOFT13_SEQ_NO     ] has no name match.

===== SEQUENCE=S01_SELECTIONS_A CurrValue=14
```

```
***** TurboIMAGE Dataset=[SELECTIONS][D ][4] Table=[SELECTIONS]:
OT_GetSetInfo - [ALBUMCODE                ] COLUMNSCnt=[0]
OT_GetSetInfo - [SELECTIONNAME            ] COLUMNSCnt=[1]
OT_GetSetInfo - [COMPOSERNAME             ] COLUMNSCnt=[2]
OT_GetSetInfo - [TIMING                   ] COLUMNSCnt=[3]
OT_GetSetInfo - [PERFORMERS               ] COLUMNSCnt=[4]
OT_GetSetInfo - [COMMENT_IMS              ] COLUMNSCnt=[5]
OT_GetSetInfo - [IMAXSOFT13_PATH_01       ] COLUMNSCnt=[6]
OT_GetSetInfo - [IMAXSOFT13_PATH_02       ] COLUMNSCnt=[7]
OT_GetSetInfo - [IMAXSOFT13_PATH_03       ] COLUMNSCnt=[8]
OT_GetSetInfo - [IMAXSOFT13_SEQ_NO        ] COLUMNSCnt=[9]

===== TABLE[SELECTIONS] Column Summary:
WARN [OT_GetSetInfo] Column[IMAXSOFT13_PATH_01      ] has no name match.
WARN [OT_GetSetInfo] Column[IMAXSOFT13_PATH_02      ] has no name match.
WARN [OT_GetSetInfo] Column[IMAXSOFT13_PATH_03      ] has no name match.
WARN [OT_GetSetInfo] Column[IMAXSOFT13_SEQ_NO       ] has no name match.

===== SEQUENCE=S01_SELECTIONS CurrValue=12

***** TurboIMAGE Dataset=[LOG][D ][5] Table=[LOG]:
OT_GetSetInfo - [ALBUMCODE                ] COLUMNSCnt=[0]
OT_GetSetInfo - [SELECTIONNAME            ] COLUMNSCnt=[1]
OT_GetSetInfo - [STARTTIME                ] COLUMNSCnt=[2]
OT_GetSetInfo - [ENDTIME                  ] COLUMNSCnt=[3]
OT_GetSetInfo - [ANNOUNCER                ] COLUMNSCnt=[4]
OT_GetSetInfo - [IMAXSOFT13_PATH_01       ] COLUMNSCnt=[5]
OT_GetSetInfo - [IMAXSOFT13_PATH_02       ] COLUMNSCnt=[6]
OT_GetSetInfo - [IMAXSOFT13_SEQ_NO        ] COLUMNSCnt=[7]

===== TABLE[LOG] Column Summary:
WARN [OT_GetSetInfo] Column[IMAXSOFT13_PATH_01      ] has no name match.
WARN [OT_GetSetInfo] Column[IMAXSOFT13_PATH_02      ] has no name match.
WARN [OT_GetSetInfo] Column[IMAXSOFT13_SEQ_NO       ] has no name match.

===== SEQUENCE=S01_LOG CurrValue=12

----- Database Integrity Checking Summary:
WARN [DBIntegrity] Table[DBACCESS                ] has no name match.
WARN [DBIntegrity] Table[DBLOCK                  ] has no name match.
WARN [DBIntegrity] Table[DBLOCK1                 ] has no name match.
WARN [DBIntegrity] Table[DBLOCK2                 ] has no name match.
WARN [DBIntegrity] Table[DOOR_SYNC               ] has no name match.

----- Checksum Verification:

<< otCHECKChecksum ORACLE_USER = [OT/JO] >>
<< otCHECKChecksum TIBASE_NAME = [MUSIC.TIDATA.IMAXSOFT] >>
<< otCHECKChecksum TIFile      = [/users/lee/musicdemo/tidb/ti] >>

Checksum Summary Report:

----------- ------- FROM DATABASE -------- --------- FROM TIFile --------
DBNAME    : MUSIC.TIDATA.IMAXSOFT          MUSIC.TIDATA.IMAXSOFT
TIFILE    : ../tidb/ti                     /users/lee/musicdemo/tidb/ti
VERSION   : A.01.04                        A.01.04
ITEM-CKSUM: 171                            171
DSET-CKSUM: 305                            305
PATH-CKSUM: 96                             96

otCHECKChecksum Result = 0
                         1 = Mactched
                         0 = Not Mactched
                        -1 = Error

Process Start: 2002-11-11 16:58:30
Process Stop:  2002-11-11 16:58:37
```

19 AMISYS only features.

- Column full or partial name replacement, such as replacing # with _NBR.
- IMAXSOFT13_SEQ_NO name change, such as using IMAGE_RECNUM instead.
- Path control column name prefix change, such as using AMISYS instead of IMAXSOFT13.
- Position all OpenTURBO internal generated columns at the beginning of the table, the default is at the end of each table.
- X-TO-NUMBER and CHAR type conversion rules.

OpenTURBO Utilities for HP-UX and ORACLE:

```
otANALYSERam
otCLEANam
otDBCOPYam
otINTEGRITYam
otSETChecksumam
otgenLOADam
tidrvam
tiloadam
```

OpenTURBO Emulator Libraries for HP-UX and ORACLE:

```
libotam.1
libotam.sl
libotdbgam.1
libotdbgam.sl
libtiam.1
libtiam.sl
```

OpenTURBO Utilities for HP e3000 and ORACLE:

```
OTAMISYS.BIN.IMAXSOFT
TIDRVAM.BIN.IMAXSOFT
TOLOADAM.BIN.IMAXSOFT
```

OpenTURBO Emulator Libraries for HP-UX and ORACLE:

```
OTAM.A.IMAXSOFT
OTAMDBG.A.IMAXSOFT
OTQAM.A.IMAXSOFT
OTQAMDBG.A.IMAXSOFT
```

20 Installation:

HP3000 Installation:

1) LEETECH account and its groups,

PUB:	AIM Middleware and utilities

2) DOOR account and its groups,

PUB:	DOOR utilities
CONFIG:	must allow (R,W,A,L,S,X:ANY) and cross account access, contains execution control and working files
LOG:	must allow (R,W,A,L,S,X:ANY) and cross account access, contains execution logfiles

3) IMAXSOFT account and its groups,

A:	OpenTURBO libraries
BIN:	OpenTURBO utilities
PUB:	OpenTURBO libraries and utilities
OTSCRIPT:	ORACLE loader script files
OTDATA:	TurboIMAGE unload data files
OTSCHEMA:	ORACLE schema
OTCMD:	OpenTURBO scripts for unload
OTWORK:	Working temp group 1
OTCOPY:	Working temp group 2
ORACLE:	Control files for OpenTURBO emulator and unload utilities
COBCNTL:	Control files for COBOLII profiler

HP9000 Installation:

1) Login root and create 3 directories,

```
mkdir /users/lee
mkdir /opt/door
mkdir /etc/opt/door
mkdir /opt/leetech
mkdir /etc/opt/leetech
```

2) Login root and create iMaxsoft's user and group

```
groupadd imaxsoft
useradd –g imaxsoft imaxsoft
groupadd lee
useradd –g lee lee
```

3) Login root, change owner and group to imaxsoft for directories /opt/imaxsoft, /opt/door, and /opt/leetech and change access permissions for /opt/imaxsoft, /opt/door, and /opt/leetech to rwxrwxr-x

```
chown imaxsoft:imaxsoft /users/lee
chown imaxsoft:imaxsoft /opt/door
chown imaxsoft:imaxsoft /opt/leetech
chown imaxsoft:imaxsoft /etc/opt/door
chown imaxsoft:imaxsoft /etc/opt/leetech

chmod u=rwx,g=rwx,a=rwx /users/lee
chmod u=rwx,g=rwx,a=rx /opt/door
chmod u=rwx,g=rwx,a=rx /opt/leetech
chmod u=rwx,g=rwx,a=rwx /etc/opt/door
```

```
   chmod u=rwx,g=rwx,a=rwx /etc/opt/leetech

4) Login imaxsoft and tar from iMaxsoft install tape.
5) /users/lee/ directory contains following sub-directories:

   ./lib/      share libraries
   ./bin/      listener and utilities programs
   ./lbin/     server programs
   ./conf/     control files, config files, etc.
   ./listner/  listener start-up and stop scripts
   ./demo/     contains directories /musicdemo, /mfcobol and
               /acucobol
   ./db        tifile
   ./lee       working directory
   ./admin     administrative tools and scripts

6) /opt/leetech/ directory contains following sub-directories:

   ./lib/      share libraries
   ./bin/      license control and utilities programs
   ./lbin/     listener, server and utilities programs
   ./demo/     sample programs
   ./share/    reserved for future include header files

7) /etc/opt/leetech/ contains license control files
   CNTLCSF.PUB.LEETECH, CNTLCSFC.PUB.LEETECH, CNTLSDK.PUN.LEETECH,
   CNTLSDKC.PUB.LEETECH, and CNTLSTBY.PUB.LEETECH

8) /opt/door/ directory contains following sub-directories:

   ./bin/      license control and utilities programs
   ./lbin/     listener, server and utilities programs

9) /etc/opt/door/ contains license control file CNTLDOOR.PUB.DOOR
```

Installation:

iMaxsoft OpenTURBO package contains 5 pieces:

DAT Tape 1 for HP e3000 OpenTURBO CORE (A.02.00)
DAT Tape 2 for HP e3000 OpenTURBO DOOR (A.02.00)
DAT Tape 3 for HP e3000 OpenTURBO AIM (A.02.00)
DAT Tape 4 for HP-UX OpenTURBO CORE (A.02.00)
CD for DBA/QUERY and Documentation

Installing on HP e3000 MPE/iX:

Installing OpenTURBO Core (Tape 1):

1. Log on as system manager:
   :HELLO MANAGER.SYS
2. Restore the TAPE 1 as follows:
   :File T;Dev=Dat or Device Number
   :Restore *T;@.@.IMAXSOFT;CREATE
3. After the tape has been restored, execute the command file to set proper access security and permissions for IMAXSOFT account and its groups:
   :INSTALL.PUB.IMAXSOFT

Installing OpenTURBO DOOR (Tape 2):

1. Log on as system manager:
   :HELLO MANAGER.SYS
2. Restore the TAPE 2 as follows:
   :File T;Dev=Dat or Device Number
   :Restore *T;@.@.DOOR;CREATE
3. After the tape has been restored, execute the command file to set proper access security and permissions for IMAXSOFT account and its groups:
   :DOORINST.PUB.DOOR

Installing OpenTURBO AIM (Tape 3):

1. Log on as system manager:
   :HELLO MANAGER.SYS
2. Restore the TAPE 3 as follows:
   :File T;Dev=Dat or Device Number
   :Restore *T;@.@.LEETECH;CREATE
3. After the tape has been restored, execute the command file to set proper access security and permissions for IMAXSOFT account and its groups:
   :LTINST.PUB.LEETECH

   Standard requirements for LEETECH, IMAXSOFT and DOOR accounts:

   1. Account access security required:
      (R,X:ANY;W,A,L:AC)
   2. Account capability required:
      AM,AL,GL,ND,SF,BA,IA,MR,DS,PH,PM
   3. PUB group access security required:
      (R,X:ANY;W,A,L,S:AC)
   4. PUB group capability required:
      BA,IA,MR,DS,PH,PM

   If you plan to initiate DOOR's interceptor and shooter processes outside the DOOR account, then you must set:

DOOR account: **ACCESS=(R,X,W,A,L:ANY)**
CONFIG.DOOR group: **ACCESS=(R,X,W,A,L,S:ANY)**

Installing on HP-UX (TAPE 4):

1. "Root" authority is necessary to create user, group, and directories,

```
groupadd imaxsoft
useradd –g imaxsoft imaxsoft
groupadd lee
useradd –g lee lee

mkdir /users/lee
mkdir /opt/door
mkdir /etc/opt/door
mkdir /opt/leetech
mkdir /etc/opt/leetech

chown imaxsoft:imaxsoft /users/lee
chown imaxsoft:imaxsoft /opt/door
chown imaxsoft:imaxsoft /opt/leetech
chown imaxsoft:imaxsoft /etc/opt/door
chown imaxsoft:imaxsoft /etc/opt/leetech

chmod u=rwx,g=rwx,a=rwx /users/lee
chmod u=rwx,g=rwx,a=rx /opt/door
chmod u=rwx,g=rwx,a=rx /opt/leetech
chmod u=rwx,g=rwx,a=rwx /etc/opt/door
chmod u=rwx,g=rwx,a=rwx /etc/opt/leetech
```

2. Mount the OpenTURBO TAPE 4 and place the drive online.
3. Restore the OpenTURBO files as follows:
   $tar -xvf /dev/rmt/0m (device name)
4. It will create the proper directories and restore files:

```
$/users/lee/lib
$/users/lee/bin
$/users/lee/lbin
$/users/lee/conf
$/users/lee/listner
$/users/lee/demo
$/users/lee/db
$/users/lee/lee
$/users/lee/admin


$/opt/leetech/bin
$/opt/leetech/demo
$/opt/leetech/lbin
$/opt/leetech/lib
$/opt/leetech/newconfig
$/opt/leetech/share
$/etc/opt/leetech                    license control files:
                                     CNTLCSF.PUB.LEETECH,
                                     etc.

$/opt/door/bin
$/opt/door/lbin
$/etc/opt/door                       license control file:
                                     CNTLDOOR.PUB.DOOR
```

Installing DBAQuery on Windows:

1. Insert OpenTURBO CD into the CD drive for installation.
2. Open folder /DBAQuery.

3. Auto-run will automatically begin installation or click on SETUP.EXE command and follow the directions on the screen.

21 Error and Warning Messages:

```
// OpenTURBO Error Messages (TurboIMAGE Emulation)
// Date: 2002/09/10
400:: GENERAL
0     = SUCCESSFUL EXECUTION - NO ERROR
-1    = NO SUCH DATABASE
-11   = BAD DATABASE NAME OR PRECEDING BLANKS MISSING
-12   = DATABASE MUST BE IN LOGON GROUP AND ACCOUNT
-13   = NOT ALLOWED; MUST BE CREATOR OF ROOT FILE OR DATABASE
-21   = BAD PASSWORD
-22   = MAINTENANCE WORD REQUIRED
-23   = USER (CLASS) LACKS WRITE ACCESS TO DATA SET
-31   = DBGET MODE ILLEGAL  FOR DETAIL DATA SET
-32   = UNOBTAINABLE ACCESS MODE
-33   = MODE 7 DIAGNOSIS NOT ALLOWED
-34   = DATABASE MUST BE RECOVERED BEFORE ACCESS IS ALLOWED
-51   = LIST TOO LONG OR NOT PROPERLY TERMINATED
-52   = ITEM SPECIFIED IS NOT AN ACCESSIBLE SEARCH ITEM IN THE SPECIFIED SET
-53   = DBPUT LIST IS MISSING A SEARCH OR SORT ITEM
-82   = CIUPDATE IS SET TO DISALLOWED; CANNOT USE CRITICAL ITEM UPDATE
-90   = ROOT FILE BAD
-99   = UNSUPPORTED FEATURE
-121  = ILLEGAL LOCK DESCRIPTOR COUNT
-123  = ILLEGAL RELATIONAL OPERATOR
-124  = DESCRIPTOR LENGTH ERROR; MUST BE 9 OR MORE
-125  = ILLEGAL SET NAME OR NUMBER IN DESCRIPTOR
-126  = ILLEGAL ITEM NAME OR NUMBER IN DESCRIPTOR
-127  = ILLEGAL ATTEMPT TO LOCK ON A COMPOUND ITEM
-128  = VALUE FIELD TOO SHORT FOR THE ITEM SPECIFIED
-129  = P28 IS LONGEST P-TYPE ITEM THAT CAN BE LOCKED
-130  = ILLEGAL DECIMAL DIGIT IN TYPE 'P' DATA VALUE
-131  = LOWERCASE CHARACTER IN TYPE 'U' DATA VALUE
-132  = ILLEGAL DIGIT IN TYPE 'Z' DATA VALUE
-133  = ILLEGAL SIGN CHARACTER IN TYPE 'Z' DATA VALUE
-134  = TWO LOCK DESCRIPTORS CONFLICT IN SAME REQUEST
-135  = DBLOCK CALLED WITH LOCKS ALREADY IN EFFECT IN THIS JOB/SESSION
-136  = DESCRIPTOR LIST LENGTH EXCEEDS 4094 BYTES
-137  = USER ABOUT To WAIT FOR SELF
-139  = INVALID NUMBER OF BASE IDs
-140  = BAD BASE ID LIST
-151  = TEXT LENGTH GREATER THAN 512 BYTES
-198  = TOTAL DBOPEN COUNT PER USER EXCEEDS LIMIT
-212  = DATABASE CORRUPTION DETECTED
-229  = CANNOT DELETE MANUAL MASTER WITH EMPTY CHAINS
-258  = INVALID ARGUMENT FOR INDEX
-259  = INVALID MODE FOR INDEX
-260  = NO PREVIOUS LIST OF QUALIFIED DATA ENTRIES
-305  = INVALID DATA SET NUMBER
-306  = INVALID DATA SET TYPE
-307  = INVALID RECORD NUMBER FOUND
-420  = FEATURE NOT IMPLEMENTED
-421  = BTE: UNKNOWN QUALIFIER VALUE FOR DBCONTROL MODE 13
-422  = BTE: DATE SET # NOT IN VALID RANGE
-423  = BTE: B-TREE ALREADY EXISTS
-424  = BTE: FAILED TO CREATE B-TREE
-425  = BTE: DB NOT OPENED EXCLUSIVELY
-426  = BTE: B-TREE DOESN'T EXIST
-429  = BTE: DBFIND ARGUMENT VERSION IS BAD
-430  = BTE: DBFIND (mode 4/24) ARGUMENT TYPE IS BAD
-431  = BTE: DBFIND (mode 4/24) ARGUMENT #1 LENGTH IS BAD
-432  = BTE: WILDCARD NOT ASCII
-433  = BTE: DBFIND (mode 4/24) ARGUMENT #2 LENGTH IS BAD
-434  = DATASET DETAIL INSTEAD OF MASTER
-436  = BTE: FAILED TO EXTRACT DATA FROM ROOT FILE
-437  = BTE: FAILED TO CONVERT @c TO [] DBFIND
-439  = BTE: CONVERSION OF KEY FROM EXTERNAL TO INTERNAL FORMAT FAILED
-444  = BTE: DBFIND ON NON-KEY FAILED OF MASTER
-446  = BTE: ARGUMENT 2 SPECIFIED FOR RELOP OF (</<=/=/>=/>)
-452  = BTE: KEY LENGTH GREATER THAN 252 BYTES (MAXIMUM INDEX KEY SIZE)
-458  = DBOPEN FAILED. OUT OF DISK SPACE
10    = BEGINNING OF FILE
11    = END OF FILE
12    = DIRECTED BEGINNING OF FILE
13    = DIRECTED END OF FILE
14    = BEGINNING OF CHAIN
15    = END OF CHAIN
16    = THE DATA SET IS FULL
17    = THERE IS NO CHAIN FOR THE SPECIFIED SEARCH ITEM VALUE
```

```
18    = BROKEN CHAIN - FORWARD AND BACKWARD POINTERS NOT CONSISTENT
20    = DATABASE CURRENTLY LOCKED SETS OR ENTRIES LOCKED WITHIN DATABASE
22    = DATA SET ALREADY LOCKED
23    = CANNOT LOCK SET DUE TO LOCKED ENTRIES WITHIN IT
24    = ENTRIES CURRENTLY LOCKED USING DIFFERENT ITEM
25    = CONFLICTING DATA ENTRY LOCK ALREADY IN EFFECT
26    = IMMINENT DEADLOCK
41    = DBUPDATE ATTEMPTED TO MODIFY VALUE OF CRITICAL ITEM: KEY/SEARCH/SORT
42    = DBUPDATE WILL NOT ALTER A READ-ONLY DATA ITEM
43    = DUPLICATE KEY VALUE N MASTER
44    = CAN'T DELETE A MASTER ENTRY WITH NON-EMPTY DETAIL CHAINS
49    = ILLEGAL BUFFER ADDRESS
50    = USER'S BUFFER IS TOO SMALL FOR REQUESTED DATA
60    = DATABASE ACCESS DISABLED
61    = PROCESS HAS THE DATABASE OPEN 63 TIMES; NO MORE ALLOWED
69    = BAD DATABASE
401:: DBOPEN
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-13   = MUST BE CREATOR OF ROOT FILE OR DATABASE.
-21   = BAD PASSWORD.
-22   = MAINTENANCE WORD REQUIRED.
-31   = BAD MODE.
-32   = UNOBTAINABLE MODE.
-34   = DATABASE MUST BE RECOVERED BEFORE ACCESS IS ALLOWED.
-90   = ROOTFILE BAD.
-94   = DATABASE BAD.
60    = DATABASE ACCESS DISABLED.
61    = THIS DATABASE OPENED MORE THAN 63 TIMES BY THE SAME PROCESS.
402:: DBINFO
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-21   = BAD DATA SET REFERENCE.
-31   = BAD MODE.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
49    = ILLEGAL BUFFER ADDRESS.
50    = BUFFER IS TOO SMALL.
403:: DBCLOSE
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-21   = BAD DATA SET REFERENCE.
-31   = BAD MODE.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
-232  = ILLEGAL DBCLOSE MODE 2 USED DURING AN ACTIVE DYNAMIC TRANSACTION.
-235  = DYNAMIC TRANSACTION ABORTED DUE TO DBCLOSE MODE 1; DATABASE CLOSED.
-420  = FEATURE NO IMPLEMENTED.
404:: DBFIND
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-21   = BAD DATA SET REFERENCE.
-31   = BAD MODE.
-51   = BAD LIST LENGTH.
-52   = BAD ITEM.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
-258  = INVALID ARGUMENT FOR INDEX.
-259  = INVALID MODE FOR INDEX.
-260  = NO PREVIOUS LIST OF QUALIFIED DATA ENTRIES.
17    = NO MASTER ENTRY.
405:: DBGET
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-21   = BAD DATA SET REFERENCE.
-31   = BAD MODE.
-51   = BAD LIST LENGTH.
-52   = BAD LIST OR BAD ITEM.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
10    = BEGINNING OF FILE.
11    = END OF FILE.
12    = DIRECTED BEGINNING OF FILE.
13    = DIRECTED END OF FILE.
14    = BEGINNING OF CHAIN/QUALIFIER ENTRIES.
15    = END OF CHAIN/QUALIFIER ENTRIES.
17    = NO ENTRY.
18    = BROKEN CHAIN.
49    = ILLEGAL BUFFER ADDRESS.
50    = BUFFER IS TOO SMALL.
406:: DBUPDATE
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-12   = NO LOCK COVERS THE DATA ENTRY TO BE ADDED.
-14   = ILLEGAL INTRINSIC IN CURRENT ACCESS MODE.
```

```
-21   = BAD DATA SET REFERENCE.
-31   = BAD MODE.
-51   = BAD LIST LENGTH.
-82   = CIUPDATE IS SET TO DISALLOWED; CANNOT USE CRITICAL ITEM UPDATE.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
17    = NO ENTRY.
41    = DBUPDATE ATTEMPTED TO MODIFY VALUE OF CRITICAL ITEM: KEY/SEARCH/SORT.
42    = READ ONLY ITEM.
49    = ILLEGAL BUFFER ADDRESS.
50    = BUFFER TOO SMALL.
407:: DBPUT
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-12   = NO LOCK COVERS THE DATA ENTRY TO BE ADDED.
-14   = ILLEGAL INTRINSIC IN CURRENT ACCESS MODE.
-21   = BAD DATA SET REFERENCE.
-23   = DATA SET NOT WRITABLE.
-24   = OPERATION NOT ALLOWED ON AUTOMATIC MASTER DATA SET.
-31   = BAD MODE.
-51   = BAD LIST LENGTH.
-52   = BAD LIST OR BAD ITEM.
-53   = MISSING SEARCH OR SORT ITEM.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
16    = DATA SET FULL.
18    = BROKEN CHAIN.
43    = DUPLICATE KEY ITEM VALUE.
408:: DBDELETE
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-12   = NO LOCK COVERS THE DATA ENTRY TO BE DELETED.
-14   = ILLEGAL INTRINSIC IN CURRENT ACCESS MODE.
-21   = BAD DATA SET REFERENCE.
-24   = DBDELETE NOT ALLOWED ON AUTO MASTER.
-31   = BAD MODE.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
17    = NO ENTRY.
44    = CANNOT DELETE MASTER ENTRY WITH NON-EMPTY DETAIL CHAINS.
409:: DBLOCK
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-31   = BAD MODE VALUE.
-121  = DESCRIPTOR COUNT ERROR.
-123  = ILLEGAL RELOP IN A DESCRIPTOR.
-124  = DESCRIPTOR TOO SHORT. MUST BE GREATER THAN OR EQUAL TO 9.
-125  = BAD SET NAME/NUMBER.
-126  = BAD ITEM NAME/NUMBER.
-127  = ATTEMPT TO LOCK USING A COMPOUND ITEM.
-128  = VALUE FIELD TOO SHORT IN A DESCRIPTOR.
-129  = P-TYPE ITEM LONGER THAN P28 SPECIFIED.
-130  = ILLEGAL DIGIT IN A P-TYPE VALUE.
-131  = LOWERCASE CHARACTERS IN TYPE U VALUE.
-132  = ILLEGAL DIGIT IN TYPE Z VALUE.
-133  = ILLEGAL SIGN IN TYPE Z VALUE.
-134  = TWO DESCRIPTORS CONFLICT.
-135  = DBLOCK CALLED WHEN LOCKS ALREADY IN EFFECT.
-136  = DESCRIPTOR LIST EXCEEDS 4094 BYTES.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
20    = DATABASE LOCKED OR CONTAINS LOCKS.
22    = DATA SET LOCKED BY ANOTHER PROCESS.
23    = ENTRIES LOCKED WITHIN SET.
24    = ITEM CONFLICTS WITH CURRENT LOCKS.
25    = ENTRY OR ENTRIES ALREADY LOCKED.
26    = LOCK NOT PERFORMED SINCE DEADLOCK WOULD OCCUR.
410:: DBUNLOCK
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-31   = BAD MODE.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
411:: DBCONTROL
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-14   = ILLEGAL INTRINSIC IN CURRENT ACCESS MODE.
-31   = BAD MODE.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
-224  = DBCONTROL MODE 1 NOT ALLOWED INSIDE A DYNAMIC TRANSACTION.
412:: DBBEGIN
413:: DBEND
414:: DBMEMO
418:: DBEXPLAIN
419:: DBERROR
420:: DBXBEGIN
```

```
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-31   = BAD (UNRECOGNIZED) DBXBEGIN MODE.
-139  = INVALID NUMBER OF BASE IDS.
-140  = BAD BASE ID LIST.
-151  = TEXT LENGTH GREATER THAN 512 BYTES.
-152  = DBXBEGIN CALLED WHILE A TRANSACTION IS IN PROGRESS.
-217  = DBOPEN MODE INCOMPATIBLE WITH DYNAMIC ROLLBACK.
-221  = CANNOT BEGIN TRANSACTION WHEN A DYNAMIC TRANSACTION IS ACTIVE.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
421:: DBXEND
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-31   = BAD (UNRECOGNIZED) DBXBEGIN MODE.
-140  = BAD BASE ID LIST.
-151  = TEXT LENGTH GREATER THAN 512 BYTES.
-217  = DBOPEN MODE INCOMPATIBLE WITH DYNAMIC ROLLBACK.
-222  = ONLY DBXUNDO ALLOWED WHEN A DYNAMIC TRANSACTION ENCOUNTERS AN ERROR.
-223  = CANNOT DBXEND OR DBXUNDO A TRANSACTION WHICH WAS NOT ACTIVE.
-238  = MDBX DBXBEGIN, DBXEND MODE MISMATCH.
422:: DBXUNDO
0     = SUCCESSFUL EXECUTION - NO ERROR
-11   = BAD DATABASE REFERENCE.
-31   = BAD (UNRECOGNIZED) DBXBEGIN MODE.
-140  = BAD BASE ID LIST.
-151  = TEXT LENGTH GREATER THAN 512 BYTES.
-223  = CANNOT DBXEND OR DBXUNDO A TRANSACTION WHICH WAS NOT ACTIVE.
-238  = MDBX DBXBEGIN, DBXEND MODE MISMATCH.
-240  = MDBX MODE MISMATCH.
```

Appendix A: ORACLE Reserved Words:

```
/************************************************************/
/* RESERVED WORDS (ORACLE)         IMAXSOFT Corp. 2002-03-09 */
/************************************************************/
/* Suffix that is attached to the end of each reserved word  */
/* The length of suffix is from 1 to 20 characters           */
/************************************************************/
suffix=_IMS
all
allocate
alter
analyze
and
any
arraylen
as
asc
at
audit
authorization
avg
begin
between
bind
both
break
by
cache
call
cast
char
character
character
charf
charz
check
close
collection
comment
commit
connect
constraint
constraints
context
continue
convbufsz
count
create
current
currval
cursor
database
date
dateformat
datelang
day
deallocate
dec
decimal
declare
default
define
delete
deref
desc
describe
descriptor
display
distinct
do
double
drop
else
enable
end
endif
escape
exec
```

```
exec
execute
exists
explain
extract
fetch
float
flush
for
force
found
free
from
function
get
global
go
goto
grant
group
having
hour
iaf
identified
ifdef
ifndef
immediate
in
indicator
input
insert
integer
intersect
interval
into
is
is
leading
level
like
list
lob
local
lock
long
max
message
min
minus
minute
mode
month
multiset
nchar
nchar_cs
next
nextval
noaudit
not
notfound
nowait
null
number
numeric
nvarchar
nvarchar2
object
ocibfilelocator
ociblobblocator
ociclobblocator
ocidate
ociextproccontext
ocinumber
ociraw
ocirowid
ocistring
of
only
open
option
option
```

```
or
oracle
order
output
overlaps
package
partition
precision
prepare
prior
procedure
put
raw
read
real
ref
reference
register
release
rename
replace
return
returning
revoke
role
rollback
rowid
rownum
savepoint
second
section
select
set
set
smallint
some
sql
sql_context
sql_cursor
sqlerror
sqlwarning
start
statement
stddev
stop
string
sum
sysdate
sysdba
sysoper
table
temporary
the
threads
time
timestamp
timezone_hour
timezone_minute
to
tools
trailing
transaction
trigger
trim
truncate
type
uid
ulong_varchar
union
unique
unsigned
update
use
user
using
uvarchar
validate
values
varchar
varchar
varchar2
```

```
variables
variance
varnum
varraw
view
whenever
where
with
work
year
zone
modify
```

Appendix B: Limitations

Data Type Mapping and Rules:

TurboIMAGE Data Type	OpenTURBO Data Type
E	Float
R	Float - Convert to IEEE format (128-bit)
I	Integer
J	Integer
P	Packed Decimal
Z	Zone Decimal - with and without sign
K	Binary
U	Binary
X	Binary

Limitations:

TurboIMAGE Constrains and Limitations:

1. Security Classes: 64 Per Database
2. Security Password Size: 8 Characters
3. Data Items: 1023 Per Data Base
4. Data Sub Item Count: 1 to 255
5. Data Sub Item Length: 1 to 255
6. Data Item Size: 2047 Half-words in length (4094 Characters)
7. Data Entries: 255 Fields Per Set
8. Data Sets: 199 Data Sets Per Database
9. Manual and Automatic Master Set: 16 Detail Sets Linkage Per Master
10. Detail Set Paths: 16 Paths Per Detail Set
11. Concurrent Access Constrains:

TurboIMAGE DBOPEN Constrains:

First User DBOPEN	First User DML	Concurrent DBOPEN	Concurrent DML	LOCK Requirements
1	ALL	1 or 5	ALL	LOCK for P,D,U
2	F,G,U	2 or 6	F,G,U	
3	ALL	-	-	DB Exclusive
4	ALL	4 or 6	F,G	DB Exclusive for P,D,U
5	F,G	1 or 5	ALL	LOCK for P,D,U
6	F,G	2, 4, 6, or 8	ALL	
7	F,G	-	-	DB Exclusive
8	F,G	6 or 8	F,G	

F – DB**F**IND, G – DB**G**ET, P – DB**P**UT, D – DB**D**ELETE, U – DB**U**PDATE
DBUPDATE doesn't allow update any detail set PATH or SORT items in modes other than 1, 3 and 4 (**CIUPDATE Permission**). DBUPDATE doesn't allow update manual master KEY item at all.

OpenTURBO Debugging Facility Levels:

```
LTDBG0   - OpenTURBO Internal Core ERROR
LTDBG1   - OpenTURBO Core Library Call Trace
LTDBG2   - OpenTURBO Reserved Words
LTDBG3   - OpenTURBO Error Messages
LTDBG4   - OpenTURBO Emulator Call Trace
LTDBG5   - OpenTURBO SDK Call Trace and CURSOR POOL Size
LTDBG6   - OpenTURBO DUAL MODE Diff Results
LTDBG7   - OpenTURBO Transaction Performance Trace
LTDBG13  - TurboIMAGE Call Flow Trace

LTDBG17  – Network Traffic Dump in Hex and Text formats
LTDBG18  – Socket Information
LTDBG19  – Net/IPC Information

LTDBG27  – Dynamic SQL Statement Preparation Trace
LTDBG28  – SQL Statement Execution Error
LTDBG29  – MALLOC, CALLOC and FREE Tracing
```

LTDBG7 - is used to locate the long transactions in your program, which can be defined via OT_TRX_THRESHOLD = n seconds, OpenTURBO reports any SQL transactions that takes longer than n seconds processing time.

LTDBG13 - is used to analyze your program's TurboIMAGE API calls; the results will be used for future SQL Code Generator. The analysis contains:

1. DBOPEN modes and concurrency.
2. DBFIND and DBGET database access methods and searching qualifiers.
3. DBLOCK and DBUNLOCK concurrency, isolations and descriptors.
4. DBXBEGIN, DBXEND and DBXUNDO for multiple databases transaction.
5. DBLOCK and DBUNLOCK for multiple databases transaction.
6. DBPUT, DBDELETE and DBUPDATE database update methods.

Appendix C: Utilities and Run Options Summary

OpenTURBO Programs on HP3000

PROGRAM NAME	DESCRIPTION	LOCATION
ENCRYPT	Encrypt a password	BIN.IMAXSOFT
DECRYPT	Decrypt a password	BIN.IMAXSOFT
DOORMAP	Generate OpenTURBO-DOOR Data Replication Mapping Rules	BIN.IMAXSOFT
GENFTP	Generate FTP Job to Transfer TurboIMAGE Data and Script Files to HP-UX	BIN.IMAXSOFT
OTDRV	OpenTURBO TurboIMAGE to Relational Transformation Core Driver	BIN.IMAXSOFT
OTLITE	OpenTURBO TurboIMAGE to Relational Transformation Core Driver (Light Version)	BIN.IMAXSOFT
TIDRV	OpenTURBO Testing Driver - supports all TurboIMAGE calls	BIN.IMAXSOFT
TILOAD	OpenTURBO Root-file Generator	BIN.IMAXSOFT
LISTNER	OpenTURBO Listener	PUB.IMAXSOFT
LANUTIL	OpenTURBO Server Process Monitor	PUB.IMAXSOFT
LTUPGRAD	Extend OpenTURBO Product License	PUB.IMAXSOFT
LTVALIDA	Check OpenTURBO Expiration Date	PUB.IMAXSOFT
INTERCOT	OpenTURBO-DOOR TurboIMAGE Updates Interceptor	PUB.DOOR
SHOOTOT	OpenTURBO-DOOR ORACLE Data Shooter	PUB.DOOR
LISTNER	OpenTURBO-DOOR Listener	PUB.DOOR
LANUTIL	OpenTURBO-DOOR Server Process Monitor	PUB.DOOR
LTUPGRAD	Extend OpenTURBO-DOOR Product License	PUB.DOOR
LTVALIDA	Check OpenTURBO-DOOR Expiration Date	PUB.DOOR

OpenTURBO Sample Scripts and Control Files on HP3000

SCRIPT NAME	DESCRIPTION	LOCATION
INSTALL	OpenTURBO Product Installation Script	PUB.IMAXSOFT
JLISTNER	OpenTURBO DUAL-MODE Listener Job	PUB.IMAXSOFT
ALL	Generate Relational Database Schema and Extract TurboIMAGE Data	SCRIPT.IMAXSOFT
SCHEMA1	Generate Relational Database Schema without Automatic Dataset	SCRIPT.IMAXSOFT
SCHEMA2	Generate Relation Database Schema with Automatic Dataset	SCRIPT.IMAXSOFT
GENFTP	Generate FTP Script	SCRIPT.IMAXSOFT
GENTI	Generate OpenTURBO Root-file	SCRIPT.IMAXSOFT
SETALL	Extract All TurboIMAGE Datasets	SCRIPT.IMAXSOFT
SDOORMAP	Generate OpenTURBO-DOOR Data Replication Mapping Rules	SCRIPT.IMAXSOFT
INSTALL	OpenTURBO-DOOR Product Installation Script	PUB.DOOR
DRSTRTOT	Start OpenTURBO-DOOR Jobs	PUB.DOOR
OTERROR	OpenTURBO Error Messages	ORACLE.IMAXSOFT
RESERVE	ORACLE Reserved Words	ORACLE.IMAXSOFT
CONFIG	OpenTURBO Emulator Configuration Control File	CONFIG.IMAXSOFT

OpenTURBO Libraries on HP3000

LIBRARY NAME	DESCRIPTION	LOCATION
DRIVER	OpenTURBO XL Library for OTDRV and	A.IMAXSOFT

	OTLITE	
OTQRY	OpenTURBO Emulator XL Library for QUERY.PUB.SYS	A.IMAXSOFT
OTQRYDBG	OpenTURBO Emulator XL Library for QUERY.PUB.SYS - DUAL-MODE, CALL ANALYZER, and Real-time Debugging	A.IMAXSOFT
OTXL	OpenTUBRO Emulator XL Library	A.IMAXSOFT
OTXLDBG	OpenTURBO Emulator XL Library - DUAL-MODE, CALL ANALYZER, and Real-time Debugging	A.IMAXSOFT
TIDRV	OpenTURBO TIDRV XL Library	A.IMAXSOFT
LTXL	OpenTURBO Middleware XL Library	PUB.IMAXSOFT
LTRL	OpenTURBO Middleware RL Library	PUB.IMAXSOFT

OpenTURBO Programs on HP9000

PROGRAM NAME	DESCRIPTION	LOCATION
Encrypt	Encrypt a password	$_OTB_ROOT/bin
Decrypt	Decrypt a password	$_OTB_ROOT/bin
otANALYER	OpenTURBO Migration Validation Reports	$_OTB_ROOT/bin
otCLEAN	Clean up any dangling keys in Automatic Table; OpenTURBO doesn't delete un-used keys off Automatic Table automatically	$_OTB_ROOT/bin
otDBCOPY	Make a copy of a full or partial logical TurboIMAGE database in ORACLE	$_OTB_ROOT/bin
otDBPURGE	Purge a logical TurboIMAGE database off ORACLE	$_OTB_ROOT/bin
otDBUTIL	Similar to DBUTIL	$_OTB_ROOT/bin
otSETChecksum	Sync logical TurboIMAGE database and TIFile checksum and version	$_OTB_ROOT/bin
otgenDB	Refine and create ORACLE specific database objects on top of OpenTURBO standard database schema	$_OTB_ROOT/bin
otgenLOAD	Refine and tune ORACLE specific data loading scripts	$_OTB_ROOT/bin
tigenSEC	Mimic TurboIMAGE access classes and passwords to ORACLE	$_OTB_ROOT/bin
tiSQLCHG	Apply TurboIMAGE alike structure changes to ORACLE	$_OTB_ROOT/bin
tiSQLCHG_PATH	Apply TurboIMAGE alike structure changes to ORACLE	$_OTB_ROOT/bin
tidrv	OpenTURBO Test Driver - supports all TurboIMAGE API	$_OTB_ROOT/bin
tidrvdbg	OpenTURBO Test Driver - with built-in debugging trace	$_OTB_ROOT/bin
tiload	Check TIFile version and re-generate TurboIMAGE schema from TIFile	$_OTB_ROOT/bin
dvqry	DBAQuery for Windows - server engine	/opt/leetech/lbin
listner	OpenTURBO Listener	/opt/leetech/lbin
lanutil	OpenTURBO Server Process Monitor	/opt/leetech/bin
ltupgrad	Extend OpenTURBO Product License	/opt/leetech/bin
ltvalida	Check OpenTURBO expiration date	/opt/leetech/bin
dbsvr	OpenTURBO ORACLE SQL core server	$_OTB_ROOT/lbin
doordbsvr.oracle	OpenTURBO-DOOR ORACLE server	/opt/door/lbin

OpenTURBO Sample Scripts and Control Files on HP9000

SCRIPT NAME	DESCRIPTION	LOCATION
all	Create and load ORACLE database	$_OTB_DEMO/script
createOTBuildDBScript	Generate ORACLE creator-user, table, index, trigger, sequence and data loading scripts	$_OTB_DEMO/script
createOTLoadDBScript	Generate ORACLE loader script and loader specific tuning	$_OTB_DEMO/script
createSECScript	Generate ORACLE security and authority to mimic TurboIMAGE access classes and passwords	$_OTB_DEMO/script
movePROD	Move OpenTURBO scripts to production directory	$_OTB_DEMO/script
otDBReport	Post database migration validation reports	$_OTB_DEMO/script

rSetCheckSum	OpenTURBO Root-file TIFile and ORACLE database checksum and version control	$_OTB_DEMO/script

OpenTURBO Libraries on HP9000

LIBRARY NAME	DESCRIPTION	LOCATION
libdbutx.1	OpenTURBO User Exit Routines Library	$_OTB_ROOT/lib
liblt.1	OpenTURBO Middleware Library	$_OTB_ROOT/lib
libot.1	OpenTURBO Emulator Library	$_OTB_ROOT/lib
libotdbg.1	OpenTURBO Emulator Library - DUAL-MODE, CALL ANALYZER, and Real-time Debugging	$_OTB_ROOT/lib
libsdk.1	OpenTURBO SQL Server Core Library	$_OTB_ROOT/lib
libsdkc.1	OpenTURBO SQL Client Core Library	$_OTB_ROOT/lib
libti.1	OpenTURBO Root-file Library	$_OTB_ROOT/lib

Option syntax: all options can be entered in tow formats, -aASCII or -a ASCII.

HP/3000 Utilities:

otdrv	tiload	ticopy	genftp	tidrv	
-a					ASCII or BINARY
-b					Base Type: ORACLE
-c					Chronological Order
-d	-d	-d	-d		Database Name
			-h		Host Name
				-i	Input Command File
			-k		Schema File Group
-l			-l		Dataset Loader Script
(-m mode)	-m			-map	Map
		-n			New DBName
-o		-o	-o	-o	Output File or Group
(-p pass)			-p		HP-UX Directory
-r LoaderDat	-r RWFile	-r RWFile	**-r LoaderDat**	-r RWFile	Loader Data File Group Reserve Word Filename
-recnum					Use TurboIMAGE RecNum
-s					Dataset Spec
-t					TranCode
-ti	-t	-t			TIFileName
-test					Internal Testing Only
			-u		HP-UX Login User
-v RWFile	-v Version				Reserve Word Filename or Version Checking
-w					Working Group
-x					Exclusive or Password

otdrv: -d **-ti -v** -t -r -l -o -w -s -c -a -x -recnum -b -test
tiload: -d -t -r -v -m
genftp: -d -r -l -k -o -h -p -u
ticopy: -d -t -r -n -o
tidrv: -r -i -o -map

HP/9000 Utilities:

tiSQLCHG	otDBCOPY	otgenLOAD	otgenDB	tigenSEC	otCLEAN	otANALYZER
		-c CACHE				
-d OldBase	-d OldBase	-d	-d	-d	-d	-d
			-f First			
			-k Schema			
	-n NewBase					
-o NewTIFile	-o NewTIFile				-o Log	-o Report
-p	-p		**-p Password**	-p		
-r Reserve	-r Reserve			-r Reserve	**-r Action**	**-r Report ID**
-s TblSpace	-s TblSpace		**-s Tblspace**	-s TblSpace		
-t OldTIFile	-t OldTIFile	**-t TABLEName**		-t TIFile		
-u	-u	-u	-u		-u	-u
-x Script	-x Script	-x Script	-x Script	-x Script		

```
tiSQLCHG:   -u -d -t -r -o -p -s -x
thDBCOPY:   -u -d -t -r -n -o -p -s -x

otgenLOAD:  -u -d -t -c -x
            (OTLoadDB_<-x Script>.EXECUTE/DROP/SQLLDR/GEN)
otgenDB:    -u -d -k -p -s -f -x
            (OTBuildDB_<-x Script>.EXECUTE/DROP)
tigenSEC:   -d -t -r -p -s -x
            (OTCreateSEC_<-x Script>.EXECUTE/sql)

otCLEAN:    -u -d -r -o
otANALYZER: -u -d -r -o
```

Appendix D - DOOR: refer to DOOR Manual in the subdirectory DOOR_MANUAL.

You may use DOOR's OpenTURBO extension utility, DOORMAP, to create MAPFile from one TurboIMAGE to ORACLE and comply with OpenTURBO emulation rules.

A. DOORMAP program creates the TurboIMAGE to ORACLE database structure mapping file, which is used by DOOR core engine for real-time repaid data replication. This is the tool that you need to keep one or more colons of your TurboIMAGE database for maintaining concurrent development environments along with migration, such as data warehousing, web application, HP-UX ORACLE migration, etc.

   You may use DOORMAP –m option to re-generate TurboIMAGE and ORACLE mapping cross reference report from –o mapfile.

   The DOORMAP program:

```
DOOR Mapping Tool Terminal Mode <A.01.02> iMAXSOFT Corp.
Copyright 2002.
run doormap.bin;info='-b -o -h -d -t -x -u -n -l –c –m –i –p'

-bDBNAME:    TurboIMAGE Fully Qualified Database Name
-oFILE  :    DOOR MAP File Name
-hHOST  :    TurboIMAGE Database Machine IP or DNS Host Name
-dHOST  :    ORACLE Database IP or DNS Host Name
-tDBTYPE:    Relational Database Vendor Name, such as ORACLE
-xTYPE  :    Destination OS Type, such as UNIX
-uUSER  :    Destination Logon User and Password
-nNAME  :    Destination ORACLE Instance Name [ORACLE:SID]
-lLOGON :    Destination ORACLE Logon User and Password
-mOUT   :    Output File for Mapping Database Structure
-c      :    Preserve Detail Dataset Path Chronological Order
             (IMAXSOFT13_PATG_nn)
-iTIFile:    OpenTURBO TIFile Name
-pPORT  :    Backend Listener's Port Number, Default=32700
```

   The DOORMAP run Script:

```
doormap.bin;info= ' &
-b MUSIC.TIDATA.IMAXSOFT &
-o cfg001 &
-h 207.92.64.66 &
-d 207.92.64.8 &
-t ORACLE &
-x UNIX &
-u lee/leepass &
-n v816' &
-l MUSIC.TIDATA.IMAXSOFT/MGR &
-c '
```

   Sample script:

```
setvar LTDBG1 1
setvar LTDBG2 0
setvar LTDBG3 0
setvar ltdbgout '$stdlist'
comment ------------------------------------------------------------
comment Create TIFile
comment ------------------------------------------------------------
purge pnrdbti.base802.peti
tiload.bin.ims0100 "-d pnrdb.baseq802.peti -t pnrdbti.baseq802.peti &
-r reserve.oracel.imaxsoft"
comment ------------------------------------------------------------
```

```
comment Create DOOR MAPFile
comment ------------------------------------------------------------
file pnrdb.baseq802.peti=pnrdbti.baseq802.peti
doormap.bin;info= ' &
-b pnrdb.baseq802.peti &
-o pnrdbdr.baseq802.peti &
-h rzq7 &
-d uxhtst01 &
-t ORACLE &
-x UNIX &
-u imaxsoft/imaxsoft &
-c -l pnrdb_baseq802_peti/pnrdb &
-n trippoc'
reset pnrdb.baseq802.peti
```

Notes:

1) All TurboIMAGE database names must be fully qualified with group and account.
2) The above script creates the OpenTURBO TIFile first, then use it for DOORMAP program to generate DOOR's MPEFile.
3) You may use –i TIFile option to specify TIFile or use MPE 'file' command as in the above example.
4) In the –l option, if you need append SQLNet string to the end of ORACLE logon, you need to surrounded it by quotes, due to special character @, it will look like –l 'pnrdb_baseq802_peti/pnrdb@trippoc'.

B. DOOR interceptor and shooter batch processes on HP3000.

DOOR interceptor (INTERCOT) reads TurboIMAGE logfile and writes to FIFO file (a MSG file), if TurboIMAGE is enabled for logging, when it hits EOF, it waits and awakes up periodically based upon your pre-set interval, otherwise it stops.

DOOR shooter (SHOOTOT) reads FIFO file, generates SQL statement and shoots the SQL statements to the target ORACLE machine for updates.

Example script for interceptor and shooter:

START802.DOOR.PETI:

```
################################################
# DOOR Interceptor and Shooter Startup Script
################################################
# USER DEFINED SECTION
#----------------------------------------------
# This section to be defined by user!
################################################
#
# Define FIFO       Name of Data Pipe (MSG File)
# Define LOGFILE    IMAGE Database Logging File
# Define DATABASE   IMAGE Dababase Name
# Define MODE       IMAGE Database OPEN Mode, Normally 1.
# Define RUN_MODE   DOOR Start Mode.
#        0 - RESTART (default) or RESUME
#        1 - START
#
setvar FIFO,       "FIFO802.LOG.PETI"
setvar LOGFILE,    "LOGPNR.LOG.PETI"
setvar DATABASE,   "PNRDB.BASEQ802.PETI"
setvar MODE,       1
setvar START,      1
setvar INTERVAL,   10
setvar FIFOSIZE,   50000
#
```

```
################################################
# End of user defined section.
################################################
#
# Note from this point on, user may only change
# LTDBG## debug flags under LeeTech's technical
# staff guidance
#-----------------------------------------------
# Set script parameter names
#
setvar jdi,    "jdi"
setvar jcard,  "JDOOR802,MGR.PETI"
setvar INTERC, "intercot.pub.door"
setvar SHTCMD, "STCMD802.DOOR.PETI"
#
# Build temporary job card file.
# Note if parms cause rec lengths > 80 use
# & to concatenate commands
#
build !jdi;rec=-256,16,f,ascii;disc=!FIFOSIZE,32;temp
#
# Create job card for DOOR Interceptor
#
echo !!JOB !jcard >>!jdi
echo !!setvar LTDBGOUT "DOOR802I.LOG.PETI" >> !jdi
#
# Set debug flags
#
# echo !!setvar LTDBG1  1  >>!jdi
# echo !!setvar LTDBG14 1  >>!jdi
# echo !!setvar LTDBG15 1  >>!jdi
# echo !!setvar LTDBG17 1  >>!jdi
# echo !!setvar LTDBG18 1  >>!jdi
# echo !!setvar LTDBG19 1  >>!jdi
#
# Remove FIFO file, if exists
#
echo !!continue          >>!jdi
echo !!purge !FIFO       >>!jdi
echo !!purge !LTDBGOUT   >>!jdi
echo !!file config.config.door=CONFIG.DOOR  >>!jdi
#
# Run DOOR Interceptor
#
echo !!RUN !INTERC &                            >>!jdi
echo !;info="!LOGFILE !DATABASE !MODE !FIFO & >>!jdi
echo !!START !INTERVAL !SHTCMD"                 >>!jdi
echo !!EOJ >>!jdi
#
# Cleanup section
#
print  !jdi
continue
file jobnum.pub.door=jnum802.LOG.PETI
purge jnum802.LOG.PETI
stream !jdi > jnum802.LOG.PETI
save jnum802.LOG.PETI
purge  !jdi,temp
```

STCMD802.DOOR.PETI

```
parm dbname, fifofile
```

```
################################################
# DOOR Shooter Startup Script
################################################
# This script may never altered by user, it is
# automatically triggered by Interceptor program.
#
# User may only change LTDBG## debug flags under
# LeeTech's technical staff guidance.
#
################################################
# Set script parameter names.
#
setvar jds,    "jds"
setvar jcard,  "JDRS802,MGR.PETI"
setvar SHOOT,  "shootot.pub.door"
setvar DRSTOP, "stopdoor.pub.door"
setvar TIFile, "PNRDBTI.BASEQ802.PETI"
#
################################################
#
# Build temporary job card file.
# Note if parms cause rec lengths > 80 use
# & to concatenate commands
#
build !jds;rec=-256,16,f,ascii;disc=50000,32;temp
#
# Create job card for DOOR Shooter
#
echo !!JOB !jcard >>!jds
#
# Set debug flags
#
# echo !!setvar LTDBG1 1  >>!jds
# echo !!setvar LTDBG13 1  >>!jds
# echo !!setvar LTDBG14 1  >>!jds
# echo !!setvar LTDBG15 1  >>!jds
# echo !!setvar LTDBG17 1  >>!jds
# echo !!setvar LTDBG18 1  >>!jds
# echo !!setvar LTDBG19 1  >>!jds
echo !!setvar LTDBGOUT "DOOR802S.LOG.PETI" >> !jds
echo !!setdump                             >> !jds
echo !!purge !LTDBGOUT                     >> !jds
echo !!file config.config.door=CONFIG.DOOR >> !jds
echo !!file jobnum.pub.door=JNUM802.LOG    >> !jds
#
# Run Shooter
#
echo !!RUN !SHOOT &      >>!jds
echo !;info="-d !dbname -l !fifofile -t !TIFile -c" >>!jds
echo !!RUN !DRSTOP;info="!fifofile 2"               >>!jds
echo !!EOJ               >>!jds
#
# Cleanup section
#
print  !jds
stream !jds
purge  !jds,temp
```

C. DOOR listener and ORACLE server on HP9000.

On HP3000, $export TWO_TASK=TRIPPOC should direct all ORACLE connection to TRIPPOC ?

You must have proper $ORACLE_HOME, $SHLIB_PATH, etc. setup in your .profile for OpenTURBO listener.

D. The global CONFIG file and CONFIG.CONFIG.DOOR working group.

The CONFIG.CONFIG.DOOR is a working group used by DOOR's programs, so it must have proper access setting for backend DOOR jobs that are outside the DOOR account. DOOR account needs ACCESS=(R,W,A,L,X:any) and CONFIG.DOOR needs ACCESS=(R,W,A,L,S,X:any).

File CONFIG.CONFIG.DOOR is used by DOOR interceptor and shooter to locate the MAPFile and TurboIMAGE related control files, you may use MPE 'file config.config.door=' command to redirect it before running interceptor and shooter.

Example CONFIG.CONFIG.DOOR:

```
PNRDB.BASEQ802.PETI,PNRDBDR.BASEQ802.PETI
REFDB.BASEREFI.PETI,REFDBDR.BASEREFI.PETI
BKODB.BASEBKO.PETI,BKODBDR.BASEBKO.PETI
```

E. –recnum option for DOOR.

ORACLE schema generation: otdrv.bin.imaxsoft –t1 or –t2
ORACLE detail dataset unload: otdrv.bin.imaxsoft –t20

Manual Master, the recnum is not used, the primary unique key is used instead (primary path's secondary migration).

Appendix E - AIM Middleware: refer to AIM User Manual in subdirectory AIM_MANUAL.

Special patch:

Current listener supports: info=, and parms=, simply add ;info='' and ;parms= to the end of your program name, such as, myprog;info='-f cntlfle';parms=4.

pri= option support:

In the listener job, now you can set your program's run priority via :setvar commands.

In the default mode, not setting anything, is CS.

:setvar LEETECH_PRG_DEFAULT_PRI "DS"

The above command sets all son processes spawned in the job in DS priority.

:setvar [your fully qualified program name]_PRI "ES"

For example :setvar MYPROG_MYGRP_MYACCT_PRI "ES", sets the specific program MYPROG.MYGRP.MYACCT in ES mode.

Appendix F: Performance and Scalability

Overall Performance Benchmarking Report 2003-01-15:

Hardware:

HP e3000 Model 928 256M Memory
HP e3000 Model 969/200 512M Memory
HP e3000 Model N-Class 2-Way 1.5G Memory
HP 9000 Model rp5470 2-Way 2G Memory

Software:

HP-MPE/iX 6.5 and TurboIMAGE
HP-UX 11i and ORACLE 9i
OpenTURBO A.01.04
OpenTURBO A.02.00 Pre-release

Performance Tuning at OpenTURBO Level for read access:

OT_IGNORE_CHAINSTATUS:

This option supersedes OT_BULKCHAINGET setting.

What's the impact to your program:

1) DBFIND return status:
   a. Half-word 5 and 6 (chain length): 0
   b. Half-word 7 and 8 (chain last recnum): 0
   c. Half-word 9 and 10 (chain first recnum): 0
2) DBGET return status:
   a. Half-word 5 and 6 (current recnum): 0
   b. Half-word 7 and 8 (backward recnum): 0
   c. Half-word 9 and 10 (forward recnum): 0
3) If your program doesn't use chain count, last and first recnum of a chain from DBFIND, and doesn't use current, backward and forward recnum from DBGET, then this is the fastest data retrieval method.
4) This option improves data retrieval performance in average about 35% - 40% compare to OT_BULKCHAINGET.

OT_BULKCHAINGET:

If OP_IGNORE_CHAINSTAUS = ON, then OT_BULKCHAINGET setting is ignored.

What's the impact to your program:

It restricts you to perform concurrent bi-directional chain retrieval. For example, you are in the middle of mode 5 DBGET, based upon certain condition, your program decides to perform a mode 6 DBGET; this is not allowed when OT_BULKCHAINGET is ON.

This option improves data retrieval performance significantly in average about 3-7 times.

Performance Analysis:

1) A native ORACLE pre-processor C/C++ program performs at OpenTURBO's OT_BULKCHAINGET level. Both of then don't support bi-directional cursor fetch, but OpenTURBO does provide full status info.

2) OpenTURBO's OT_IGNORE_STATUS offers the best performance, in average is about 20%-30% slower than native TurboIMAGE, but is also the most restricted one. It provide almost no status info.
3) ORACLE init<sid>.ora:
    1. SORT BUFFER SIZE
    2. POOLS SIZE - java and data
    3. Log Files
    4. Archive and Audit options
4) ORACLE Table Spaces:
    1. Data and Index
    2. Large vs. Small Tables
    3. ANALYZE ... COMPUTE STATISTICS;
    4. EXPLAIN PLAN and HINT

29 OTPCOB – COBOLII External Call Parameter Boundary Checker.

- OTPCOB is a HP3000 program; it invokes MPE/XL COBOLII to compile your program, and analyzes your program's external calls and their associated parameters to ensure they are on 4-byte word boundary, which is not enforced in COBOLII.
- OTPCOB is written in HP3000 C, it is not ANSI compliant due the MPE/XL C backward compatibility (refer to source code directory, C for source, H for header files and M for make).

OTPCOB.BIN.IMAXSOFT:

OpenTURBO OTPCOB <A.02.00> iMaxsoft Corp. Copyright 2002.

run otpcob.bin.imaxsoft;info='-a MyProg.make.acct'

-aMAKEFile: COBOLII to MF-COBOL Analysis Only.
MAKEFile contains instructions to compile your COBOLII program.
1) you must add $CONTROL MAP into your cobol source code.
2) you must setup all file equations for copylib and include files.

You must add $CONTROL MAP at beginning of COBOLII source file MyProg.source.acct.

COBOLII Compilation Makefile MyProg.make.acct:

```
file coblib=COBLIB.source.acct
cob85xl MyProg.source.acct
```

The MAKEFILE is a MPE/XL command file that contains all necessary commands to compile a COBOLII program. It is in standard TEXT file format and is un-numbered.

In you COBOLII program, you must add $CONTROL MAP compilation option at the beginning of the file to inform compiler to generate a SYMBOL TABLE MAP in your compilation list-file.

Example:

How to run OTPCOB.BIN.IMAXSOFT and its options:

```
:setvar LTDBG1 1
:setvar LTDBG2 0
:setvar LTDBG4 0
:setvar LTDBGOUT '$stdlist'
:FILE VERBS.COBCNTL=VERBS.COBCNTL.acct
:FILE DELMTER.COBCNTL=DELMTER.COBCBTL.acct
:otpcob.bin.imaxsoft;info="-a MyProg.make.acct"
```

1. LTDBG1 set to 1 (ON) to get standard ERROR and WARN messages.
2. LTDBG2 should be 0 (OFF), unless you need to see the entire symbol boundary information.
3. LTDBG4 should be 0 (OFF), unless you want to trace OTPCOB internal logic.
4. LTDBGOUT redirects OTPCOB output to a file or $stdlist, depends upon your setting.

Sample Output:

OTPCOB: COBLST - External Call's PARM BOUNDARY Report:

```
ERROR: SUB=HPEXTIN                          PARM=PI-ERROR
ERROR: ==> PI-ERROR is OFF BOUNDARY[DP+1794] LINE=[440].
       Line 440, declares parameter PI-ERROR for routine "HPEXTIN"
       in TEKTC001 is not on 4-byte boundary, DP+1794 % 4 = 2, so
       it is on 2-byte boundary.

ERROR: SUB=DBOPEN                           PARM=IMAGE-PASSWORD
ERROR: SUB=DBCONTROL                        PARM=IMAGE-PASSWORD
ERROR: ==> IMAGE-PASSWORD is OFF BOUNDARY[DP+26] LINE=[1143].
       Line 1143, declares parameter IMAGE-PASSWORD for routines
       "DBOPEN" and "DBCONTROL" in TKETC001 is not on 4-byte
       boundary, DP+26 % 4 = 2, so it is on 2-byte boundary.

. . . . .

ERROR: SUB=DBFIND                           PARM=IMAGE-SEARCH-ITEM
ERROR: ==> IMAGE-SEARCH-ITEM is OFF BOUNDARY[DP+66] LINE=[1148].

ERROR: SUB=DBFIND                           PARM=IMAGE-ARGUMENT
ERROR: SUB=DBGET                            PARM=IMAGE-ARGUMENT
ERROR: ==> IMAGE-ARGUMENT is OFF BOUNDARY[DP+82] LINE=[1149].

ERROR: SUB=DBOPEN                           PARM=IMAGE-STATUS
ERROR: SUB=DBCONTROL                        PARM=IMAGE-STATUS
ERROR: SUB=DBCLOSE                          PARM=IMAGE-STATUS
ERROR: SUB=DBFIND                           PARM=IMAGE-STATUS
ERROR: SUB=DBGET                            PARM=IMAGE-STATUS
ERROR: SUB=DBPUT                            PARM=IMAGE-STATUS
ERROR: SUB=DBUPDATE                         PARM=IMAGE-STATUS
ERROR: SUB=DBDELETE                         PARM=IMAGE-STATUS
ERROR: SUB=DBLOCK                           PARM=IMAGE-STATUS
ERROR: SUB=DBUNLOCK                         PARM=IMAGE-STATUS
ERROR: SUB=DBXBEGIN                         PARM=IMAGE-STATUS
ERROR: SUB=DBXEND                           PARM=IMAGE-STATUS
ERROR: SUB=DBXUNDO                          PARM=IMAGE-STATUS
ERROR: SUB=DBERROR                          PARM=IMAGE-STATUS
ERROR: SUB=DBEXPLAIN                        PARM=IMAGE-STATUS
ERROR: SUB=DBINFO                           PARM=IMAGE-STATUS
ERROR: ==> IMAGE-STATUS is OFF BOUNDARY[DP+182] LINE=[1152].

ERROR: SUB=DBOPEN                           PARM=IMAGE-MODE
ERROR: SUB=DBCONTROL                        PARM=IMAGE-MODE
ERROR: SUB=DBCLOSE                          PARM=IMAGE-MODE
ERROR: SUB=DBFIND                           PARM=IMAGE-MODE
ERROR: SUB=DBGET                            PARM=IMAGE-MODE
ERROR: SUB=DBPUT                            PARM=IMAGE-MODE
ERROR: SUB=DBUPDATE                         PARM=IMAGE-MODE
ERROR: SUB=DBDELETE                         PARM=IMAGE-MODE
ERROR: SUB=DBLOCK                           PARM=IMAGE-MODE
ERROR: SUB=DBUNLOCK                         PARM=IMAGE-MODE
ERROR: SUB=DBINFO                           PARM=IMAGE-MODE
ERROR: ==> IMAGE-MODE is OFF BOUNDARY[DP+202] LINE=[1185].

. . . . .
```