Eloquence: HP 3000 IMAGE Migration

By Michael Marxmeier, Marxmeier Software AG

Michael Marxmeier is the founder and President of Marxmeier Software AG (www.marxmeier.com), as well as a fine C programmer. He created Eloquence in 1989 as a migration solution to move HP250/HP260 applications to HP-UX. Michael sold the package to Hewlett-Packard, but has always been responsible for Eloquence development and support. The Eloquence product was transferred back to Marxmeier Software AG in 2002. This chapter is excerpted from a tutorial that Michael wrote for HPWorld 2003. Email: mike@marxmeier.com

Eloquence is a TurboIMAGE compatible database that runs on HP-UX, Linux and Windows.Eloquence supports all the TurboIMAGE intrinsics, almost all modes, and they behave identically. HP 3000 applications can usually be ported with no or only minor changes. Compatibility goes beyond intrinsic calls (and also includes a performance profile.) Applications are built on assumptions and take advantage of specific behavior. If those assumptions are no longer true, the application may still work, but no longer be useful.

For example, an IMAGE application can reasonably expect that a DBFIND or DBGET execute fast, independently of the chain length and that DBGET performance does not differ substantially between modes. If this is no longer true, the application may need to be rewritten, even though all intrinsic calls are available.

Applications may also depend on external utilities or third party tools. If your application relies on a specific tool (let's say, Robelle's SUPRTOOL), you want it available (Suprtool already works with Eloquence). If a tool is not available, significant changes to the application would be required. Eloquence goes a long way to make sure that not only the intrinsic calls are compatible, but also reasonable expectations are met. We are working with hp 3000 solution providers to make sure that your familiar environment is available.

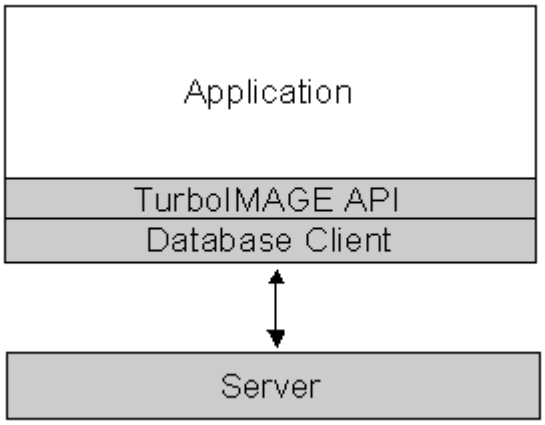
What is not supported with TurboIMAGE compatibility mode in Eloquence:

- DBCONTROL modes which are specific to TurboIMAGE implementation details
- Item level security
- Required changes: Eloquence requires the database name is terminated with a space, semicolon or NULL character

Eloguence's TurboIMAGE compatibility is implemented at different levels:

- The database server implements functionality at the backend

- The database client and utilities provide support for TurboIMAGE functionality
- The TurboIMAGE compatibility API implements source code compatibility



The TurboIMAGE compatibility API is implemented as a library on top of the native Eloquence database API. It does not impose a performance impact. The TurboIMAGE compatibility API provides source code compatibility with existing applications and translates TurboIMAGE intrinsic calls to the Eloquence API. This involves different alignment requirements, modes and status codes, emulating different behavior or converting arguments as required.

The Eloquence image3k library implements the TurboIMAGE intrinsics, and it is this library that the application (or language runtime) is linked against. If you are coding in C or C++, the image3k.h include file provides the function prototypes you will need.

Using Eloquence with AcuCOBOL

Link the Eloquence image3k library to the ACU COBOL runtime (runcbl). Then load the Eloquence image3k library dynamically (using CALL). Eloquence currently uses native byte order. On little endian platforms (Intel IA-32), COMP-5 type must be used instead of COMP. The –D5 compiler option maps all COMP to COMP-5.

Using Eloquence with Micro Focus Cobol

Link the Eloquence image3k library to the application. Eloquence currently uses native byte order. On little endian platforms (Intel IA-32), COMP-5 type must be used instead of COMP. A Micro Focus COBOL compiler directive may be used to map the COMP to the COMP-5 type.

```
MAKESYN "COMP-5" = "COMP"
```

Real World Migration Issues

Data set capacity

Since Eloquence data sets are dynamic and grow as required, data set capacity has a different meaning in Eloquence, because Eloquence has no concept of a data set specific capacity. When you ask for the capacity of a data set, Eloquence returns the highest record number allocated for a data set as the capacity value (DBINFO modes 202 and 205).

Problem: An application may check for "enough room" in a data set (which you cannot do on Eloquence, since there is no concept of a fixed capacity).

Solution: Remove or disable the capacity check.

Workaround: Return a "HUGE" value as the capacity (e.g., trap Eloquence DBINFO 202 and 205 modes and return an application-specific "capacity" value).

Don't lie to the schema

TurboIMAGE does not really care what you put in a character field, but Eloquence relies on the schema type information. Eloquence may need to convert strings to different encoding, or may need to do a byte order conversion, and uses indexes that require type specific ordering.

Solution: Use separate fields for different information, or use the correct item type.

Workaround: Use Eloquence on a single platform, or use Eloquence binary item type 'B'.

Character set encoding

On MPE, designers often use the HP-ROMAN8 character set encoding for text field data. HP-ROMAN8 encoding is typically not available on other platforms. Therefore, Eloquence defaults to the HP-ROMAN8 character set on HP-UX (and MPE) and to the ISO-8859-1 character set on other platforms. Eloquence performs conversion "on the fly".

Byte order

PA-RISC and Itanium (with HP-UX) use big endian byte order. Intel IA-32 and Itanium (Linux and Windows) use little endian byte order. Eloquence performs conversion "on the fly" if necessary.

Parameter alignment

TurboIMAGE requires most arguments to be 16 bit aligned. Eloquence relaxes most alignment restrictions. Eloquence does not require a specific alignment for string arguments

Record numbers

Eloquence uses a different algorithm to assign and reuse record numbers. TurboIMAGE uses a LIFO (last in first out) order to reuse deleted records (unless HWMPUT is active). Eloquence uses a FIFO (first in first out) order to use available record numbers. Eloquence does not support HWMPUT, application has no control over record number usage. HWMPUT is the setting that enables DBPUT to place entries at the high-water mark first, instead of at the delete chain head.

Problem: a DBDELETE / DBPUT sequence on Eloquence likely results in a different record number for the entry.

Solution: Fix the application so that it does not expect record numbers to remain inviolate over a DBDELETE/DBPUT.

Workaround: Use DBUPDATE mode 2 (same as DBUPDATE mode 1 and CIUPDATE) to do a critical item update. This does not result in a new record number.

Identical database names

Problem: TurboIMAGE supports the use of the same database name in different file groups. Eloquence requires a unique database name per server instance.

Solution: Use multiple server instances (eg. test / production environments), or add the group name to the database name (eg. DBNAME.GROUP).

Access to database files

TurboIMAGE databases reside in the file system. Applications could use file system operations to copy databases. But Eloquence databases reside in the volume files and are not accessible separately.

Solution: Copy the whole database environment or use dbstore to extract a single database and dbrestore to restore the database to another server instance. Or use dbexport / dbimport.

Moving Your Databases from TurboIMAGE to Eloquence

Schema files are compatible and no change is required. Eloquence includes MPE tools to export the database content to flat files. Then use FTP to transfer the schema file and the export files to the target system. On the target system run the

schema processor, the dbcreate utility and the dbimport utility. Data migration from TurboIMAGE to Eloquence is a straightforward process.

Install the DBEXPORT utility on the HP3000

Eloquence comes with the dbexport and dbimport utilities which can be used to unload a database to one or multiple text files or respectively load a text file into the database. The export files are text files which follow a simple syntax.

An equivalent DBEXPORT utility is available for the HP3000 and can be used to unload your database. It can be downloaded from the Eloquence ftp server

DBEXPORT is used to export the database contents to one or multiple text files. It provides a convenient means to move your database contents to the Eloquence database. The current version should be used with Eloquence A.07.00 and supports all item types.

When running from the POSIX shell the arguments are separated by a space:

```
$ DBEXPORT -p SECRET -v TESTDB
```

When running from the MPE shell (CI) you need to enclose the arguments in quotes:

```
: DBEXPORT "-p SECRET -v TESTDB"
```

DBEXPORT creates a separate file for each dataset. The name is database.set.exp (for example: "SAD.03.exp"). An automatic data set or an empty data set is ignored.

Transferring the files

Transfer your schema file and the export files to the Eloquence system, into the directory where you wish to create the database. When transferring by ftp, use text mode to transfer the schema file and use binary mode to transfer the export files.

Creating the database:

Run the Eloquence schema processor

```
$ dbschema schemafile
$ schema -T schemafile
```

Option -T selects TurboIMAGE compatibility mode.

```
$ dbcreate database
```

Import the data

Use dbimport to load the database

```
$ dbimport -v database
```

DBIMPORT finds all the database export files and imports them. For example, if the database name is SAD, it looks for SAD.03.exp, SAD.04.exp, etc.

The option -v displays the import progress. On the Windows and Linux platform you should specify the -z roman8 option to indicate the source data uses the HP-ROMAN8 encoding. This makes sure any national characters ("Umlaute") are converted.

That's it – your migration is complete.

Eloquence Product Components

The Eloquence product consists of multiple components. Besides the database, it comes with a complete software development and runtime environment and different options for user interfaces.

The Eloquence product includes the following major components:

- The Eloquence programming language, based on HP Business Basic.
- The Eloquence database management system, based on IMAGE, which we are focusing on in this chapter.
- Different user interface options (Text, Windows, Java and Web based user interfaces).
- Various development tools, including a graphical development environment on the Windows platform.

A quick product overview: There are about 2500+ installations of Eloquence worldwide. It is used by about 60+ VARs / ISVs. Installations cover a wide range of sizes from a single user to a few hundred concurrent users. In the past, Eloquence was typically used to implement vertical and customer specific solutions

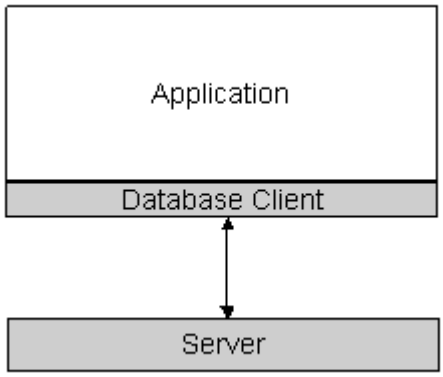
Solutions based on Eloquence include ERP, Order Management, Material Management, Financial Accounting / Payroll, Civil Service, Banks and Financial Services. Typically, Eloquence customers are small to medium sized companies that use Eloquence-based solutions for business critical tasks, such as production planning, material management or order management.

Eloquence Database Architecture

The Eloquence database is almost 100% compatible to TurboIMAGE at the application level. However, the underlying architecture is different. While the Eloquence database is almost 100% compatible with TurboIMAGE at the application level, the underlying concepts are different. This section explains some of the approaches taken by the Eloquence database and explains some of the architectural differences from TurboIMAGE.

Client/Server architecture

The Eloquence database uses a client/server-based architecture. The database server, on behalf of the application, performs all database access. The application is linked with a client library, which provides the database API (Application Programming Interface) and submits any requests that cannot be processed locally to the server and returns the results and any status codes to the application.



The figure above shows an application with the database-programming interface, which is connected to the database server.

The connection between the client side (application) and the server is made through the network. In case the application is running on the same system as the server, shared memory is used for communication between client and server to reduce the communication overhead.

When a database is opened, the server uploads some structural information about the database to the client, so the client side is able to verify the call arguments and convert them as necessary. It also allows the client side to process most DBINFO calls locally.

Network transparent

Applications running on different machines and operating systems can access a common database. Requests and results are translated transparently.

The Eloquence database is network transparent; applications running on different machines and operating systems can access a common database. The database server and client API make sure that requests and results are translated as necessary to match the specific platform requirements.

For example: PA-RISC systems use a different byte order than Intel IA32 based systems. And MPE systems use a different character set encoding. The Eloquence database client will translate any values transparently as required.

Multiple platforms

Eloquence is available for multiple operating systems and hardware architectures:

- HP-UX
- Linux
- Windows NT/2000/XP
- Database client library on MPE (not yet released)

The current version supports HP-UX on PA-RISC and Linux and Windows NT/2000/XP on the Intel IA32 architecture. Future releases will add support for the IA64 (or IPF) architecture for the HPUX, Linux and Windows operating systems.

Indexing

The Eloquence database comes with integrated indexing capabilities and Eloquence uses indexes rather then hashing with master sets. Indexes are available to applications to implement indexed database access, such as partial key retrieval or ordered retrieval.

The TurboIMAGE compatibility extension allows an application to make use of indexes in a TurboIMAGE compatible manner, since Eloquence implements a commonly used subset of the TPI functionality

With Eloquence B.07.00 both the TPI programming interface as well as TurboIMAGE indexes (used with master sets, "super-chains") are supported.

Locking

Locking with the Eloquence database is fully compatible with TurboIMAGE but provides additional options.

The most visible difference is that locking is optional and the application is free to implement the locking strategy that suits best to its requirements. The Eloquence database neither depends on explicit locking (DBLOCK) nor does it impose restrictions on locking (like requiring a set lock for specific operations).

Instead of verifying if a write operation (such as DBPUT) is covered by a DBLOCK, Eloquence verifies if a conflicting DBLOCK has been granted which covers the same data. In this case a database status is returned. This behavior is fully compatible since in TurboIMAGE all write operations must be covered by a DBLOCK and consequently there can be no concurrent DBLOCK.

Transactions

The Eloquence database relies on transactions to ensure data consistency. Transactions in Eloquence are not specific to a database. All databases modified in a transaction are part of the transaction.

Eloquence does not expose incomplete transactions to concurrent processes. Changes in a transaction are only visible to the initiating process until the transaction is committed. This is called transaction isolation. If concurrent process accesses database content modified in an incomplete transaction the original content is returned.

Transactions are not limited in size: Incomplete transactions that overflow the cache are stored in the log volume and are only limited by the disk space available in the log volume. All database procedures (such as DBPUT) are internally considered a transaction (implicit transactions) which are committed automatically once the call is completed. For example, a DBPUT can consist of 50+ internal operations and may need to be aborted cleanly at any time. We use the transaction mechanism for this.

Nested transactions are supported: If an application makes use of transactions (explicit transactions) any internal transactions become sub-transactions and are merged on commit with the upper transaction level. Transactions can also be nested in applications and then behave the same way.

Database names

The database name is no longer restricted to 6 characters and can include dots and hyphen characters in addition to letters and digits.

Eloquence databases do not reside (logically) in the file system but are managed by a server process. A database name does not specify a file location, but addresses a specific server instance. Eloquence uses an extended format for database names. The database name can specify the network address of the database server as an option. To access a database on a remote system, simply add the server address to the database name. For example:

```
[[hostname][:service]/]database
```

`Hostname` specifies database server system.

`Service` specifies database server instance.

The hostname or IP address qualifies the system on which the database server resides. The default is the local system (localhost or IP address 127.0.0.1).

The service name or port number is used by the server process on that machine. The default is defined by the tcp service named eloqdb (which defaults to port 8102 during installation).

If the hostname or service is specified, it is separated with a slash from the database name.

The example below specifies the database SAMPLEDB on the local system, using the default database server instance in different forms:

```
localhost:eloqdb/SAMPLEDB
:eloqdb/SAMPLEDB
SAMPLEDB
```

Use the EQ_DBSERVER environment variable to specify the default server instance. For example:

```
EQ_DBSERVER=invent9k.external.hp.com:8102
```

This specifies that the specified server instance manages the database. The default is used unless you provide more specific information.

Database security

The database server maintains a list of users. Database access privileges are assigned to groups, which are similar to IMAGE user classes. Then a user is made a member of one or more groups.

For each database, groups define the access rights. This is the equivalent of the TurboIMAGE user classes, which can be defined in the PASSWORDS section in the schema file. The schema processor creates two default groups "public" and "dba".

The group "dba" provides administrative access to the server and all databases, but does not allow read access to the data (it does have the right to erase a data set).

The group "public" public allows read/write access to the data, but is not authorized to perform administrative tasks (it has the access rights defined for user class 0).

Other groups are created and provide the access rights defined in the schema file. Users can be associated with up to eight groups and get the summary rights associated with the groups. Unless associated with a group, no access to the database is possible. By default, the schema processor associates the user "dba" with the group "dba" and the user "public" with the group "public" and all other groups defined in the schema file. This can be changed with the dbutil utility.

In order to connect to the Eloquence database server a user name and password is required. The new DBLOGON procedure may be used to specify user and password. A file can be specified as the user name or password. A default user is used if no specific user is specified.

The EQ_DBUSER and EQ_DBPASSWORD environment variables may be used to specify the default user or the password. For example:

```
EQ_DBUSER=file:/home/mike/dblogon
EQ_DBUSER=mike
EQ_DBPASSWORD=file:/home/mike/passwd
```

Database environment

A database environment consists of a configuration file, one or more data volumes, and a transaction log volume. Multiple database environments can coexist on the same machine, each managed by a separate server process.

Volume files

The Eloquence database does not reside in the file system but uses volume files, which reside in the file system as a container for the data and structural information. Volume files are a storage container managed by the database server. All volumes combined define the disk space accessible to the database server. The maximum size of a single volume file is 128 GB. As of Eloquence B.07.00 a single volume file is limited to 2 GB on the HP-UX and Linux platform (we currently don't use 64 bit file operations on HP-UX and Linux to maintain compatibility with older operating system versions). On the Windows platform we support 64 bit file operations so the 2 GB limit for a single volume does not apply.

A maximum of 255 volume files is supported by a single server instance, which defines an upper limit of about 500 GB for the current version and up to 32 TB once the 2 GB limit has been removed.

Volume files are specific to the system architecture (byte order).

Server catalog

Eloquence does not use a ROOT file. Database structural information is maintained in the database environment, called the server catalog. The catalog is an internal database maintained by the server process. The server catalog is created and initialized with the dbvolcreate utility (which is used to create the primary data volume) and maintained with database utilities, such as the schema processor or the dbutil utility. The dbdumpcat utility allows reading the server catalog.

Database limits – the Eloquence B.07.00 Image limits:

- 2048 data items
- 500 data sets
- 64 / 16 paths
- Entry length 5120 bytes

The Eloquence B.07.00 database either matches or exceeds the TurboIMAGE schema limits. Most TurboIMAGE limits do not apply to Eloquence because the underlying architecture is different, and Eloquence allocates most resources dynamically.

Although the Eloquence database may exceed the TurboIMAGE limits in some cases, you should be careful before making use of that fact. Applications may need to be reviewed, if they support the increased limits. As the IMAGE intrinsics do not specify buffer sizes (this is especially important with DBINFO as it returns a variable sized result) this may result in application crashes due to a buffer overflow. With the Eloquence TurboIMAGE compatibility API, DBINFO mode 406 returns information about database schema size.

Disk space within the volume files is allocated dynamically. A single data set can span multiple volumes and is therefore not limited to the size of a single volume file. Transactions can grow dynamically and are only limited by available disk space in the log volume(s).

Scalability

Database / data set size is limited by the disk space allocated to the database environment. – Current limit is ~500 GB. Hard limit is ~32 TB

Some Eloquence limits apply to a database environment rather then a specific database as all databases managed by a database environment share common resources. The size of a database or data set is only limited by the disk space available in the volume files.

The maximum disk space which can be allocated to a database environment is currently limited to about 500 GB. This is imposed by the current limitation of 2 GB per volume file.

The maximum amount of disk space, which can be managed by a database environment, is about 32 TB, which should be sufficient for a while. The number of concurrent users per database environment is limited to 1000. However we do not recommend using more than 500 concurrent users per database environment with the B.07.00 release.

Database Utilities

The database utilities can be categorized into three groups:

- Offline utilities which can only be used when the database server process is not active
- Administrative utilities which control server operation or request server status information
- Database utilities that access the database server

Offline utilities

- dbvolcreate / dbvolextend / dbvolchange / dblogreset - database volume management
- dbvoldump - display volume properties
- dbfsck - volume consistency check and simple repair tool
- dbcfix – database consistency check and repair tool
- dbrecover - forward recovery

Database utilities in the offline category operate on the volume files and can not be used while the database server is active

Administrative utilities

- dbctl - server management utility
- HTTP status monitor

Administrative utilities connect to the database server to control server operation or request server status information.

The Online Database utilities

- schema - Schema processor
- dbcreate / dberase / dbpurge - create / erase / purge database
- dbtables - database cross reference
- prschema - re-create schema from database
- dbdumpcat - catalog information utility
- dbexport / dbimport - export/import data base content to/from text file
- dbinfo - information on database tables
- dbutil - structural maintenance and database security management
- QUERY utility (different from QUERY/3000)

The online utilities make use of the database server to access a database.

Installation and Configuration of the Eloquence database

Evaluation license

By default the "Personal Edition" license key is installed. A temporary license key can be created during installation. A temporary license key can be requested from the Eloquence web site by filling out the temporary key request form

```
http://www.hp-eloquence.com/license/demo.html
```

Create eloqdb user/group

Create a user name and a group name e.g. eloqdb to be used as the owner/group of the database files. On Windows the system account is used by default. Create the user and group eloqdb. This user and group are used by the database server and own the database volume files. Administration of volume files should be performed either from root or this user. If this user is not used for administration it should be disabled.

Configure kernel parameters

Depending on the configuration, Eloquence has specific requirements for the kernel configuration. On Unix and Linux, Eloquence can use shared memory for communication, which requires additional kernel resources.

Some of the HP-UX kernel parameters that need to be configured are semaphore-related parameters, shared memory-related parameters, and process data size.

To configure kernel parameters, start SAM and select Kernel Configuration -> Configurable Parameters, change the kernel parameters as necessary and build a new kernel. Changing kernel parameters requires a system reboot.

Kernel: Semaphore configuration (EnableIPC enabled)

If you enable shared memory communication in the database server configuration file (EnableIPC), the kernel SYSV semaphore-related kernel parameters would likely need revision. For each local connection, a separate semaphore is needed. The semmni, semmap, semmns, semmnu and semume kernel parameters must be adapted accordingly.

If shared memory communication is not enabled, the semmns kernel parameter needs only be changed if a large number of I/O threads are used. The default number of 4 I/O threads is likely covered by the default kernel configuration.

In the following sections, the variable x specifies the number of concurrent connections (Threads configuration item) and y the number of i/o threads (IOThreads configuration item).

Semaphore configuration for EnableIPC=1

- Set the 'semmni' to at least x+20
- Set the 'semmap' to 'semmni' + 2
- Set the 'semmns' to at least x+y+20
- Set the 'semmnu' to at least x+20
- Set the 'semume' to at least x+20

If shared memory communication is enabled in the database server configuration file, the kernel SYSV shared memory-related kernel parameters might need revision. When EnableIPC=1 for each local connection, a separate shared memory segment is needed.

The shmmni and shmseg kernel parameters must be adapted accordingly. This is not required with EnableIPC=2. If shared memory communication is not enabled (or EnableIPC=2), the shared memory related kernel parameters do not need to be changed . In any case, the maxdsiz kernel parameter should be changed to allow at least 128 MB of process data size.

In the following sections the variable x specifies the number of concurrent connections (Threads configuration item).

Shared memory configuration

- Set the 'shmmni' to at least x+20
- Set the 'shmseg' to at least x+20

Data size

- Set the 'maxdsiz' to at least 0x08000000 (128MB)

Setup database environment

The database environment (which is a server instance) consists of

- Server configuration file (eloqdb.cfg)
- Primary data volume
- Transaction log volume(s)
- Additional data volume(s) as required

The Server configuration file

The default server configuration file is /etc/opt/eloquence6/eloqdb6.cfg

This file defines server properties, including configuration, scaling and tuning parameters, and volume files. For example, here is a simple server configuration:

```
[Server]
Service = eloqdb
ServiceHTTP = 8103
UID = eloqdb
GID = eloqdb
EnableIPC = 1
SyncMode = 0
[Config]
Threads = 100
IOThreads = 4
BufferCache = 64
CheckPtSize = 50
```

The Service configuration item specifies the service name (as defined in the services file) or the port number, which is used by the database server. The default value is eloqdb. The port number must be different for each server instance and may not already be used.

The ServiceHttp configuration item specifies the service name (as defined in the services file) or the port number used by the embedded web server. By default the HTTP status is disabled.

The Uid and Gid configuration items specify the user and group, which are used to run the server and own the volume files. This setting is ignored on Windows.

The EnableIPC configuration item activates the use of shared memory for communication. This significantly reduces system overhead but requires additional kernel configuration, as described earlier.

Shared memory: EnableIPC

- EnableIPC=0 (default) disables use of shared memory communication
- EnableIPC=1 enables use of shared memory on HP-UX and Linux

- EnableIPC=2 enables use of a single shared memory segment for HP-UX (recommended)

The SyncMode configuration item specifies if the server should write committed transactions to disk immediately. If disabled, a more efficient write strategy is used. SyncMode is enabled by default, which makes the database resistant in case of a system crash, however it causes additional disk load. A more efficient option is to enable forward logging.

Sync/Async mode

- SyncMode=1 (default) pushes all committed transactions to disk immediately and waits for completion
- SyncMode=0 (recommended) writes changes to disk asynchronously and does not wait for completion

The [Config] section defines the scaling and tuning parameter settings for the database server. This example configuration supports 100 concurrent users using a 64MB cache. You should at least have 128 MB memory on your system.

The Threads configuration item specifies the maximum number of connections. Each connection is associated with a thread running in the server context. Only a single connection is used by an application to access any number of databases managed by a single server (connection pooling). So, Threads defines the maximum number of concurrent connections for this server instance

The IOThreads configuration item defines the maximum number of concurrent I/O operations. The default is 4. This number may need to be increased depending on database size or number of concurrent users. The maximum usable setting also depends on the I/O capabilities of the server.

The BufferCache configuration item specifies the cache size in megabytes. The cache is used to reduce the number of disc accesses. The default (and min.) cache size is 5 MB.

The CheckPtSize configuration item specifies the amount of disk space used for the transaction journal. The default size is 10 MB. When this space is exhausted the server performs an internal checkpoint operation.

Create the Volume Files

```
dbvolcreate /var/opt/eloquence6/data01.vol
dbvolextend –t log /var/opt/eloquence6/log.vol
dbvolextend –t data /var/opt/eloquence6/data02.vol
```

The HP Eloquence database does not reside in the file system but uses volume files that reside in the file system as a container for the data and structural information. Please note that you need both a data volume and a log volume in order to start the data base server.

These commands should either be used by the system administrator (root) or executed by the eloqdb user.

The dbvolcreate utility is used to create a new server instance and creates the primary data volume, which contains the system catalog. The dbvolextend utility extends a database environment by another volume. This could either be a transaction log volume or another data volume. The volume type must be specified. The specified volume files are created and then added to the server configuration file. A single volume file is currently limited to 2 GB on HP-UX and Linux and additional data volumes may be required.

Configure Server Startup

Configure automatic startup of the Eloquence database. The startup configuration file defines which Eloquence services are started

- HP-UX: /etc/rc.config.d/eloquence6
- Linux: /etc/sysconfig/eloquence6

The Eloquence eloqsd service is often not needed and should not be started. Set the START_ELOQSD variable to 0 to disable the automatic start of the eloqsd service.

Starting the database server

On HP-UX:

```
/sbin/init.d/eloq6 start|stop|status|restart [instance …]
```

On Linux:

```
/etc/init.d/eloq6 start|stop|status|restart [instance …]
```

Other server operations:

- start – start server processes
- stop – stop server processes
- status – check status of server processes
- restart – restart server process

Troubleshooting

The Eloquence database writes diagnostic messages to the syslog:

- P-UX: /var/adm/syslog/syslog.log
- Linux: /var/log/messages
- Windows: application event log

Linux installation

Eloquence uses the RPM package manager. RedHat Linux 7.x to 9 and SuSE Linux 7.x to 8.x have been certified. Other Linux distributions may be used but additional manual configuration may be required.

For installation or update, execute the command below:

```
$ rpm -U Eloquence-B0700.rh8.i386.rpm
```

Note: the temporary license option is not available during installation on Linux.

Windows installation

Eloquence uses the Microsoft Installer engine that was introduced with Windows 2000. The MS installer engine is installed or updated before proceeding with the Eloquence installation (may require a reboot). The Microsoft Installer engine provides a common installation procedure across applications. The installer engine makes use of an .msi database, which specifies the product; the product files are contained in .cab files. Note: Different setup programs are used for Windows 2000/ XP/2003, Windows NT and Windows 9x.

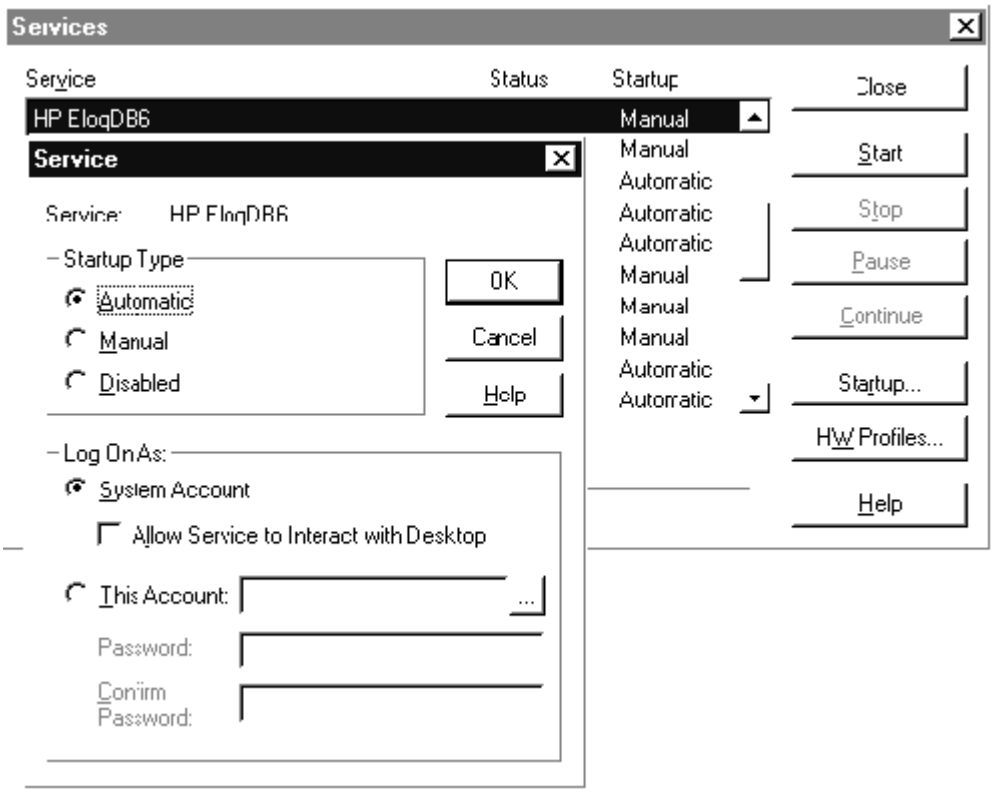
The setup program differs for the download and CD-ROM media version. The download version cannot be installed from a read-only media, as it requires write access to the local directory. The CD-ROM version is already unpacked and does not require a writeable directory.

Automatic Start

During installation, the Eloquence database is registered as a service. To configure automatic server startup, please open the service control panel (eloqdb6 service) and configure the service to start automatically.

After creating the database environment, start the service manually. In case the service startup fails, please check the windows application event log for any messages. Note: The eloqsd service is often not needed and should not be started.

See a sample Windows service configuration screen below.



Database Backup

There are two supported backup strategies: off-line and on-line, with a related option: Forward logging.

Off-line backup has 3 steps:

1. Shutdown the eloqdb6 server process
2. Backup all volume files
3. Re-start the server process

And On-line backup has 4 steps:

1. Enable on-line backup mode
2. Backup the data volume file(s)
3. Backup of the log volume is optional
4. Disable on-line backup mode

On-line backup

In on-line backup mode, the Eloquence database makes sure the data volume(s) are consistent and not changed, even if the database is in use. All database modifications are saved temporarily in the transaction log volume. This makes sure any backup software can be used to create a consistent backup without interrupting ongoing work and allows for easy and straightforward integration into backup procedures. Since the database environment is backed up, the backup will cover all databases, which are managed in that environment.

When the backup mode is finished, Eloquence copies the pending changes from the transaction log volume to the data volume(s).

The dbctl utility is used to enable on-line backup mode. Here is an example backup script:

```
$ dbctl -u file:/root/credentials backup start
$ tar -cf /dev/rmt/0m /database
$ dbctl -u file:/root/credentials backup stop
```

The database is put in on-line backup mode by sending the backup start request with the dbctl utility. Then you use the tar utility (or your favorite backup tool) to create a backup of the database environment and finally, you finish the backup by sending the backup stop command.

Forward logging

Use forward logging to record all modifications since a previous backup. It is fast and involves only minimal processing.

The server provides the option to log all changes since a previous (on-line or off-line) backup, which can then be applied with the dbrecover utility. Forward recovery is integrated with the on-line backup and Eloquence is able to discover automatically, if and which part of the log file applies to a recovered backup. As an option, the database server can manage forward recovery logging automatically. Removal of old log files can easily be integrated into the backup procedure.

Forward logging is enabled in the server configuration

```
[ForwardLog]
FwLog = /path/to/fwlog-%N.log
```

More information

For more details on Eloquence, visit the Eloquence web site at

```
http://www.hp-eloquence.com
```