HP Eloquence A.05.10 Release Notes

HP Part No. B1367-90004
Printed in Germany
02/96

First Edition

HEWLETT PACKARD

Legal Notices

Manual Organization

This manual documents all changes incorporated into HP Eloquence since the last printed manual edition included with HP Eloquence release A.04.00. It also replaces the release notes included with HP Eloquence A.05.00.

This manual is organized into the following chapters:

Chapter 1	Installation of HP Eloquence
Chapter 2	Configuration of HP Eloquence
Chapter 3	Changes related to the HP Eloquence language
Chapter 4	Changes related to the HP Eloquence database system
Chapter 5	Changes related to the HP Eloquence DLG system
Chapter 6	The HP Eloquence Dialog Manager
Appendix A	Summary of changes incorporated into HP Eloquence by version.
Appendix B	Summary of fixed problems incorporated into HP Eloquence by version.

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

1

Installing HP Eloquence

This chapter covers the installation of the HP Eloquence software.

- Installation on HP-UX
- The HP Eloquence license scheme
- Installation on PC

Installation on a HP-UX system

HP Eloquence A.05.10 can only be installed and run on the HP-UX Operating System release 9.0 or later.

HP Eloquence is installed and updated by the usual operating system tools.The installation tools depend on the operating system version.

HP-UX 9.x	/etc/update
HP-UX 10.x	/usr/sbin/swinstall

The following sections describe the procedure to install or update the HP Eloquence software in detail for each HP-UX version.

NOTE: HP Eloquence no longer needs its own installation or update tools, as it is now installed and updated by the usual operating system tools.

Updating from HP-UX 9.x to HP-UX 10.x

If you install HP Eloquence on HP-UX 9.x and update to HP-UX 10.x, you should reinstall the HP Eloquence software using swinstall. HP Eloquence expects its configuration files in the directory /etc/opt/eloquence when running on HP-UX 10.x instead of /opt/eloquence/etc when running on HP-UX 9.x. It also requires the existence of the directory /var/opt/eloquence. Reinstalling HP Eloquence with swinstall does ensure the appropriate configuration. Your configuration files are copied automatically by the installation program to the new location. You should delete them afterwards manually from /opt/eloquence/etc.

HP-UX version 9.X

As “superuser” follow the steps below to install the HP Eloquence software.

❒ Insert the HP Eloquence DDS tape into your tape drive and extract the software from tape to a temporary location.

```
cd /tmp
tar xf /dev/rmt/0m B136XB.updt
```

where /dev/rmt/0m is your DDS tape device file.

❒ Start “update(1m)” by typing:

```
/etc/update
```

To install the HP Eloquence software, follow the directions below:

- Select “Change Source or Destination ->”.
- Select “From Tape Device to Local System ...”.
- Tab to the “Source” field and enter the following: /tmp/B1368XB.updt
- Press “Done”.
- Select “Select/View Partitions and Filesets ...”
- Activate either the **B1367B** partition to install the HP Eloquence Runtime environment or the **B1368B** partition to install the HP Eloquence Development environment.
- Select “Start Loading”.
- Type “y” to “Start loading filesets now ?”.
- Check /tmp/update.log to make sure installation completed with no error.

HP-UX version 10.X

As “superuser” follow the steps below to install the HP Eloquence software.

❒ Insert the HP Eloquence DDS tape into your tape drive and extract the software from tape to a temporary location.

```
cd /tmp
tar xf /dev/rmt/0m B136XB.sd
```

where /dev/rmt/0m is your DDS tape device file.

❒ Start “swinstall(1m)” by typing:

```
/usr/sbin/swinstall -s /tmp/B136XB.sd
```

To install the HP Eloquence software, follow the directions below:

- In the Software Selection Window, highlight to select either the **B1367B** product to install the HP Eloquence runtime environment or the **B1368B** product to install the HP Eloquence development environment.
- Then choose the “Mark for Install” item from the Actions Menu. The “Marked?” column will be set to “Yes”.
- Select the “Install (analysis...)” item from the Actions Menu. When the analysis is finished with no error, i.e. Status:Ready, click OK.
- Click Yes in the Confirmation window to begin the actual installation.
- When the installation is completed, a dialog is displayed to notify you that the install task is completed. You may exit then.

HP Eloquence license scheme

The HP Eloquence Software is copy protected. In order to use it, you need a license key. Unless you already have a valid HP Eloquence license (for example, if installing a product update), a temporary license key is created during the installation process, which will expire after 30 days. In order to receive a permanent license key, you must fill out the included license form and send it to your Customer Response Center.

The location of the HP Eloquence license file depends on operating system revision:

HP-UX 9.x **`/opt/eloquence/etc/licence`**

HP-UX 10.x **`/etc/opt/eloquence/licence`**

The license file is a plain text file which contains all licences which apply to the HP Eloquence product. The utility **`/opt/eloquence/bin/chklic`** may be used to check the licence file.

After you received your permanent license key from Hewlett-Packard, you must add it to your license file in order to activate it.

NOTE: HP Eloquence A.05.10 uses a new licence scheme. Instead of "branding" the executable files with a system id, a license key is required. The license key is created automatically, if you update to HP Eloquence A.05.10.

Example license file contents

```
B1368B A.05.10 none 0-0-0 96-1-31 0 4facc94a9986447397371939a3519a62
```

Please refer to /opt/eloquence/newconfig/config/licence for a detailed description of the license file format.

Installation on PC

This section describes the installation of HP Eloquence A.05.10 client software on the PC platform.

There are two groups of software:

- Runtime software

  This software is required to use HP Eloquence A.05.10 client/server services on your PC.

- Development software

  This software has been contributed to enable you to build your own programs using HP Eloquence remote database services or to customize the HP Eloquence dialog server. This software is not installed by default.

PC platform prerequisites

Hardware requirements

- 32 Bit processor of type 80386 or above.

  Recommended is a 486 processor and a graphics accelerator card.

- at least 4 MB RAM, recommended is 8 MB RAM
- Hard disk with at least 8MB of available disc space

System software

- MS WINDOWS 3.1 or above
- MS WINDOWS 95
- MS WINDOWS NT 3.5 or above

Network software

A TCP/IP protocol stack providing the WINSOCK 1.1 API is required in order to use the HP Eloquence client/server functionality. This is included with MS Windows 95 and MS Windows NT 3.5.

NOTE: “Wolverine”, Microsoft’s TCP/IP-32 for Windows for Workgroups can be found at ftp://ftp.microsoft.com/bussys/Clients/WFW/. Microsoft have released version 3.11b. Users of version 3.11a should upgrade as bugs have been found.

Development system

To develop your own database applications using the HP Eloquence remote database access, or to customize the HP Eloquence DLGSRV program, you need the following development software:

- Microsoft Visual C/C++ 1.5

Installation

The installation involves the following steps.

- Transfer of the software from the HP 9000 to the PC.
- Creating an installation medium
- Running the setup program

Transferring pcclnt.exe

HP Eloquence does not provide a separate installation medium for the PC. The client software is installed along with the HP Eloquence software on the HP 9000 server system. The client software is provided in the file **`pcclnt.exe`** in the directory **`/opt/eloquence/share/dist`**.This file is a self extracting archive.

You can use ftp to transfer the client software to your PC. A sample ftp session is shown below:

```
$ ftp host
Connected to host ...
Name (host:unknown): login
Password: password
ftp> bin
ftp> cd /opt/eloquence/share/dist
ftp> get pcclnt.exe
ftp> bye
```

where host is the host name or IP address of your server system, login is your login name and password is the corresponding password. The transferred file is copied into the current directory.

NOTE: If transferring **`pcclnt.exe`** with ftp, please remember to set the file type to *binary*.

Creating an installation disk

In order to create an installation media, you must extract the archive file to either a floppy disc or into a directory on your hard disc. To extract the archive file execute the following command:.

```
pcclnt.exe a:
```

where **a:** is the target drive or directory.

This will extract HP Eloquence files for the PC platform to the given target path.

NOTE: If extracting the archive file into a directory on your hard disk, the directory must be named **disk1**. This is required by the setup program.

Start the setup program

Start the HP Eloquence setup program by executing

```
a:\setup
```

from within the MS Windows environment where *a:* is the path to the installation drive or directory.

The setup program provides the following choices:

- Install runtime software

  This is enabled by default. This will transfer the HP Eloquence runtime files for the PC platform to your disc and install a program group in the Windows program manager.

- Install EloqDB development files

  This choice is disabled by default. If enabled, the setup program will install the files for developing applications using HP Eloquence remote database access on the PC platform.

- Install DLGSRV development files

  This choice is disabled by default. If enabled, the setup program will install the files for customizing the HP Eloquence DLGSRV program.

Using the Set Location push button, it's possible to specify a different installation directory.

The following paths can be specified:

- Runtime files

  This specifies the location, where HP Eloquence runtime files are to be installed.

- Dialog files

This specifies the location, where HP Eloquence dialog files are to be installed.

- Windows files

  This specifies the location, where Windows related files should reside. In a shared windows installation, this should point to your local WINDOWS directory.

- HP Eloquence database development files

  This specifies the location, where HP Eloquence database development files are to be installed.

- Dialog Server development files

  This specifies the location, where dialog server development files are to be installed.

2

Configuring HP Eloquence

This chapter covers the configuration of the HP Eloquence software:

- HP-UX configuration
- Customizing HP Eloquence
- PC-related configuration

Configuration Overview

Configuring HP Eloquence includes the following steps:

1 *Configuring the HP-UX Operating System*

2 *Configuring the User Environment*

3 *HP-UX Network Configuration*

4 *Customizing the HP Eloquence Configuration Files*

5 *Starting the Daemon Process*

The HP Eloquence configuration files are located in the following directories:

```
HP-UX 9.x:   /opt/eloquence/etc
HP-UX 10.x:  /etc/opt/eloquence
```

The default configuration files are located in **`/opt/eloquence/newconfig/config`**.

If you plan to use the HP Eloquence graphical user interface, you should also:

6 *Customize the ELOQ.INI File on the HP-UX System*

If you want to use PCs running Microsoft Windows as front ends to the graphical user interface, the following additional configuration steps are required:

7 *Setup your Network Connection on the PC*

8 *Customize the ELOQ.INI File on the PC Platform*

Configuring the HP-UX Operating System

The steps in this section can only be performed by a "superuser".

In order to use HP Eloquence on the HP 9000, it may be necessary to modify some HP-UX system parameters using the menu driven SAM utility.

Listed below are the HP-UX semaphore and file system parameters along with the values they should contain. Please note that these are minimum requirements. If a higher value is already configured, a modification is not required. The NUSERS value referenced below is the value in the HP Eloquence configuration file.

Semaphores:	**Value**
semmns	min. NUSERS + 50 (default = 128)
semume	20 (default = 10)
semmnu	min. NUSERS + 30 (default = 30)
File System:	**Value**
ninode	min. NUSERS * 30
nfile	NUSERS * 80 + 100
nflocks	NUSERS * 20 + 50

NOTE: A system reboot is required to activate the new kernel.

NOTE: If you use other products that require semaphores, you should increase the above values accordingly.

Configuring the User Environment

The steps in this section can only be performed by a “superuser”.

There are two files where the user environment can be configured:

- The environment defined in **`/etc/profile`** affects all users.
- The file **`.profile`** located in a user’s home directory configures the user-specific environment.

Perform the following steps:

1 Make the HP Eloquence executables accessible:

  - If you are using HP-UX 10.x, this step is already done during software installation. The installation script automatically adds **`/opt/eloquence/bin`** to the global path configuration file **`/etc/PATH`**.
  - If you are using HP-UX 9.x, edit the file **`/etc/profile`**. This file contains the command **`PATH=`**, where you should append **`:/opt/eloquence/bin`**.

2 If you plan to frequently use the HP Eloquence migration tools, you might additionally make the **`/opt/eloquence/tools`** directory accessible.

  - If you want to make the migration tools accessible for all users and you are using HP-UX 10.x, edit the file **`/etc/PATH`** and append **`:/opt/eloquence/tools`** to the global path list.
  - If you want to make the migration tools accessible for all users and you are using HP-UX 9.x, edit the file **`/etc/profile`** and append **`:/opt/eloquence/tools`** to the **`PATH=`** command.
  - If you want to make the migration tools accessible to several users only, edit the files **`.profile`** in the appropriate home directories and append **`:/opt/eloquence/tools`** to the **`PATH=`** command.

3 Make sure that the terminal type is configured properly for all users. Normally, **`/etc/profile`** provides a mechanism that automatically detects the terminal type and sets the **`TERM=`** environment variable appropriately.

4 Make your configuration changes active. This is done by logging off and back on again.

HP-UX Network configuration

If you want to connect PCs via Network to use them as dialog clients or for remote database access, you should configure the HP-UX network appropriate.

To setup your network connection you change the following network software configuration files:

/etc/hosts	maps IP addresses and host names
/etc/services	maps service names and port numbers

Change your hosts file

Add the name of each client PC to your **`/etc/hosts`** file. Make an entry like below:

```
192.1.1.100   clientname
```

Where *192.1.1.100* is the IP address of a client PC and *clientname* is the host name of a client PC.

Change your services file

To provide a service name / port number mapping, you change your **`/etc/services`** file.

Add lines like below to your **`/etc/services`** file:

```
eloq      8000/tcp     # HP Eloquence remote database
runsrv    8010/tcp     # HP Eloquence RUNSRV utility
dmclnt    8011/tcp     # HP Eloquence dmclnt
```

Where

eloq is the service name used by HP Eloquence for remote database access.

runsrv is the service name used by the HP Eloquence **`RUNSRV`** utility.

dmclnt is the service name used by the HP Eloquence dialog server. This service defines the first port in a range of ports used by the HP Eloquence dialog server.

Verify carefully that the chosen port numbers (8000,8010,8011 in the example above) are unique.

Customizing the HP Eloquence Configuration Files

This discussion assumes that the HP Eloquence software has already been installed on the HP 9000 Series 800. The information in this section is directed to the system administrator for the HP Eloquence software.

Before HP Eloquence can be started, it must be configured. There are three sample configuration files associated with this process—**`d.eloq.config`**, **`d.group`** and **`d.eloqrc`**. These three files are located in the directory **`/opt/eloquence/newconfig/config`**. Of the three files, **`d.eloq.config`** *must* be copied and edited to conform with the environment in which it will be used. The other two configuration files, **`d.group`** and **`d.eloqrc`**, are optional, but if used they must also be copied and edited.

d.eloq.config

`d.eloq.config` is a sample global configuration file. It must be copied or renamed to a file named **`eloq.config`** and edited appropriately. This new file, **`eloq.config`**, defines the HP Eloquence environment in which all HP Eloquence users work. It contains specifications for such things as the number of users, volume names, and printer numbers. If this file is not present, the eloqd program cannot be started (please refer to *Starting the Daemon Process*).

Listed below are the steps required to set up a global configuration file:

1 Change to the configuration directory as follows:

```
HP-UX 9.x:   cd /opt/eloquence/etc
HP-UX 10.x:  cd /etc/opt/eloquence
```

2 Make a copy of the file **`/opt/eloquence/newconfig/config/d.eloq.config`** and name it as follows:

```
cp /opt/eloquence/newconfig/config/d.eloq.config eloq.config
```

3 Using the HP-UX vi editor (any editor can be used) change the file **`eloq.config`** to meet your requirements. Start vi as follows:

```
vi eloq.config
```

Use the explanatory information within the **`eloq.config`** file to aid you in this step. In addition, refer to the section entitled *The User, Group and Global Configuration Files* later in this chapter.

Changes made to the file **`eloq.config`** become active when the eloqd program is started. Therefore, if eloqd is already active when you make the changes to **`eloq.config`**, it must be killed (kill) and restarted (eloqd). eloqd is the daemon process program and is discussed in *Starting the Daemon Process*.

A listing of the **`d.eloq.config`** is included at the end of this section.

d.group (optional)

`d.group` is a sample group configuration file. It must be copied or renamed to a file named **`group.`*`group name`*** and edited appropriately. This new file, **`group.`*`group name`***, defines the HP Eloquence environment in which users associated with the named group work.

Listed below are the steps required to set up a group configuration file:

1 Change to the configuration directory as follows:

```
HP-UX 9.x:   cd /opt/eloquence/etc
HP-UX 10.x:  cd /etc/opt/eloquence
```

2 Make a copy of the file **`/opt/eloquence/newconfig/config/d.group`** and name it as follows:

```
cp /opt/eloquence/newconfig/config/d.group group.group name
```

Replace the parameter *group name* with the appropriate name.

3 Using the HP-UX vi editor (any editor can be used) change the file **`group.`*`group name`*** to meet your requirements. Start vi as follows:

```
vi group.group_name
```

Use the explanatory information within the **`group.`*`group_name`*** file to aid you in this step. In addition, refer to the section *The User, Group and Global Configuration Files* later in this chapter.

A listing of the **`d.group`** is included at the end of this section.

d.eloqrc (optional)

Setup of this file is normally done by the user, rather than the system administrator of the HP Eloquence software. Therefore, this discussion assumes the setup is performed by the user from the home directory.

`d.eloqrc` is a sample user configuration file. It must be copied or renamed to a file named **`.eloqrc`** and edited appropriately. This new file, **`.eloqrc`**, defines the HP Eloquence environment in which an individual user works.

Listed below are the steps required to set up a user configuration file:

1 Copy the file **/opt/eloquence/newconfig/config/d.eloqrc** to a file named **.eloqrc** in your home directory, as follows:

```
cp /opt/eloquence/newconfig/config/d.eloqrc .eloqrc
```

2 Using the HP-UX vi editor (any editor can be used) change the file **.eloqrc** to meet your requirements. Start vi as follows:

```
vi .eloqrc
```

Use the explanatory information within the **.eloqrc** file to aid you in this step. In addition, refer to the next section *The User, Group and Global Configuration Files*.

A listing of the **d.eloqrc** is included at the end of this section.

New HP Eloquence A.05.10 configuration directive

HP Eloquence A.05.10 supports the following new configuration directive:

LOCK CONFLICTING ITEM

Syntax:

```
LOCK CONFLICTING ITEM {0|1}
```

The LOCK CONFLICTING ITEM configuration directive makes it possible to configure, how predicate locks with conflicting items are resolved.

Former HP Eloquence revisions rejected a predicate lock with a conflicting item, because this could lead to a situation, where two processes own a lock on an overlapping subset of data. If LOCK CONFLICTING ITEM is set to 1, predicate locks with conflicting items are granted. However any write attempt to data where another process owns a lock will result in a status error -12.

The User, Group and Global Configuration Files

When HP Eloquence starts it reads the user configuration file (**.eloqrc**) first, the group configuration file (**group.*group_name***) second, and the global configuration file (**eloq.config**) third. Remember that the user and group configuration files are optional; therefore, if they are not present, the global configuration file is the first and only configuration file read.

It is important to understand how the statements within the three configuration files relate to one another. The statements pertinent to this discussion are MSI, VOLUME, PRINTER, and PORT.

MSI

An MSI (Mass Storage Is) statement may be included in the user and group configuration files. Upon start-up, HP Eloquence takes the first MSI statement it reads as the default mass storage device. This means if you have a user and a group configuration file and they both contain an MSI statement, HP Eloquence will use the MSI defined in the user configuration file.

If an MSI statement is not found, HP Eloquence uses the first VOLUME defined in the global configuration file as the default mass storage device.

VOLUME

VOLUME labels *must* be unique. If two VOLUME statements have the same label, an error is returned.

HP Eloquence recognizes all defined VOLUMES. In other words, it appends the VOLUME commands from each of the configuration files.

PRINTER

PRINTER numbers and PORT numbers *must not* conflict with one another. If this occurs, an error is returned.

As with VOLUMEs, HP Eloquence recognizes all defined PRINTERs. In other words, it appends the PRINTER commands from each of the configuration files. However, if two PRINTER statements refer to the same printer number, the first PRINTER statement read is recognized, the other statement is ignored.

NOTE: Printer numbers 8, 9, and 10 are automatically defined and reserved for the following:

8	Display terminal.
9	Bit bucket.
10	Local terminal printer.

PORTS

PORT numbers and PRINTER numbers *must not* conflict with one another. If this occurs, an error is returned.

As with VOLUMEs and PRINTERs, HP Eloquence recognizes all defined PORTS. In other words, it appends the PORT commands from each of the configuration files.

Default Configuration Files

d.eloq.config

```
#******************************************************************
#
```

```
#   HP ELOQUENCE global configuration
#   (C) Copyright Hewlett-Packard GmbH, 1988-1996
#   @(#)$Revision: 12.0 $
#
#   This file contains global available configuration
#   It must be named eloq.config and located at:
#    HP-UX 9.x    /opt/eloquence/etc
#    HP-UX 10.x   /etc/opt/eloquence
#
#   PLEASE NOTE:
#   You MUST define at least one volume (typically SYSTEM,
#   see below) or the eloqd process will not come up.
#
# Shared memory configuration
#
#      Format: <token> value
#
#      NUSERS - maximum number of simultaneous sessions
#               that can access the shared memory
#               limits:  1 .. 400
#               default: 30
#
#      NTASKS - maximum number of available secondary tasks
#               limits:  0 .. NUSERS - 1
#               default: 10
#
#      NDB    - maximum number of open databases per system
#               limits:  1 .. 100
#               default: 16
#
#      NOPEN  - maximum number of databases accesses
#               limits:  1 .. 10000
#               default: 5 * NUSERS
#
#      NLOCKS  - maximum number of locks
#               that can be active at a time
#               limits:  1 .. 10000
#               default: 3 * NOPEN
#
#      NLOCKM - maximum amount of memory that can be used by
#               predicate locks at a time (in kbytes)
#               limits:  1 .. 500
#               default: 10
#
#      NFILES  - maximum number of different (database-) files
#                that can be accessed at a time
#                limits:  1 .. 10000
#                default: 20 * NUSERS
#
#       NBUFFS - number of database buffers (in kbytes)
#                limits:  250 .. 1000
#                default: 10 * NUSERS (min 500)
#
#       DATE    - setup date format
#                 MM/DD/YY -> american format (default format)
#                 DD.MM.YY -> european format
#
# The following configuration directive does allow concurrent
# locking with "conflicting" items. However, if a conflict
# exists, any write attempt will result in a -12 status error.
#
```

```
#       LOCK CONFLICTING ITEM 1
#
# Volume configuration -> global accessible volumes
#
#      Format: VOLUME label device path
#
#      label  - Volume label (up to 8 characters)
#                 must be unique
#      device - HP260 device specifier eg. :F2,6,0
#                 must be unique
#      path   - HP-UX path to map volume on
#
# Printer configuration -> global accessible printers
#
#      Format: PRINTER no [model] type spec
#
#      no     - printer select code (-2 .. 7, 11 .. 99)
#       model   - PCL or OTHER
#                 this field is optional and for future use
#      type   - printer type PIPE or FILE
#      spec   - path/command to process on printer selection
#
# Port configuration -> global accessible ports
#
#       Format: PORT no spec
#
#       no      - port select code (11 .. 20)
#                 may not conflict with PRINTER
#       spec    - path of tty devicefile
#
#***************************************************************

# --- sample shm configuration

NUSERS 30                # (default value)
NLOCKS 200               # locks reduced from 450 to 200
NFILES 400               # isam files reduces from 600 to 400
LOCK CONFLICTING ITEM 0  # normal lock behaviour

# --- date format

DATE   "DD.MM.YY"# european date format

# --- sample volume configuration

VOLUME SYSTEM  :Z2,7,0 /opt/eloquence/share/prog
VOLUME EXAMPLE :Z2,7,1 /opt/eloquence/share/example
#VOLUMECONTRIB :Z2,7,2 /opt/eloquence/share/contrib

# --- sample global printers

PRINTER 0  PIPE "lp -s 2>/dev/null"
#PRINTER 1  PIPE "lp -s -oc 2>/dev/null"
#PRINTER 2  FILE /dev/lp

# --- sample PORT

#PORT 11 /dev/tty0p5
```

d.group

```
#*****************************************************************
#
#   HP ELOQUENCE group configuration
#   (C) Copyright Hewlett-Packard GmbH, 1989-1995
#   @(#)$Revision: 11.0 $
#
#   This file will allow adaption to whole group
#   It must be named group."group" and located at:
#    HP-UX 9.x     /opt/eloquence/etc
#    HP-UX 10.x    /etc/opt/eloquence
#
#       Format: <token> value
#
#       MSI    - Default device
#                If not specified this is the 1st global device.
#               May be specified either as label or as device string.
#
# Volume configuration -> group accessible volumes
#
#       Format: VOLUME label device path
#
#       label  - Volume label (up to 8 characters)
#                   must be unique
#       device - HP250/HP260 device specifier eg. :F2,6,0
#                   must be unique
#       path   - HP UX path to map volume on
#
# Printer configuration -> group accessible printers
#
#       Format: PRINTER no [model] type spec
#
#       no     - printer select code (-2 .. 7, 11 .. 99)
#         model   - PCL or OTHER
#                   this field is optional, default is PCL
#       type   - printer type PIPE or FILE
#       spec   - path/command to process on printer selection
#
#       Group printer are scanned in front of the global table.
#       Group entries MAY REPLACE global entries.
#
# Port configuration -> local accessible ports
#
#         Format: PORT no spec
#
#         no      - port select code (11 .. 20)
#                   may not conflict with PRINTER
#         spec    - path of tty devicefile
#
#       Group ports are scanned in front of the global table.
#       Group entries MAY REPLACE global entries.
#
#*****************************************************************

# --- group MSI volume

MSI PRVOL

# --- group devices

VOLUME PRVOL   :L2,1,0 /opt/payroll/prog
VOLUME DBVOL   :L2,1,1 /opt/payroll/db
```

d.eloqrc

```
#*****************************************************************
#
#  HP ELOQUENCE - user configuration
#  (C) Copyright Hewlett-Packard GmbH, 1988-1995
#  @(#)$Revision: 11.0 $
#
#  This file will allow adaption of users.
#  It must be located at $HOME and named .eloqrc
#
#      Format: <token> value
#
#      MSI    - Default device
#               If not specified the group MSI specicification or
#                  the 1st global device will be used.
#              May be specified either as label or as device string.
#
# Volume configuration -> local accessible volumes
#
#      Format: VOLUME label device path
#
#      label  - Volume label (up to 8 characters)
#                  must be unique
#      device - HP250/HP260 device specifier eg. :F2,6,0
#                  must be unique
#      path   - HP UX path to map volume on
#
# Printer configuration -> local accessible printers
#
#      Format: PRINTER no [model] type spec
#
#      no     - printer select code (-2 .. 7, 11 .. 99)
#       model    - PCL or OTHER
#                  this field is optional, default is PCL
#      type   - printer type PIPE or FILE
#      spec   - path/command to process on printer selection
#
#      Local printer are scanned in front of the global
#      and group table.
#      Local entries MAY REPLACE global and group entries.
#
# Port configuration -> local accessible ports
#
#       Format: PORT no spec
#
#       no      - port select code (11 .. 20)
#                 may not conflict with PRINTER
#       spec    - path of tty devicefile
#
#      Local ports are scanned in front of the global
#      and group table.
#      Local entries MAY REPLACE global and group entries.
#
#*****************************************************************

# --- MSI volume

MSI TEST
# alternate MSI ":M2,4,0"
```

```
# --- local devices

VOLUME TEST    :T2,4,0 $HOME/test

# --- local printer

PRINTER 0  PIPE "lp -dlj -onb -s 2>/dev/null"
PRINTER 1  PIPE "lp -dlj -onb -s -oc 2>/dev/null"
#PRINTER 2  FILE /dev/tty0p3
```

Starting the Daemon Process

The HP Eloquence daemon process coordinates HP Eloquence activities such as configuration and database access. The process must be started before you can use HP Eloquence.

Starting and shutting down the HP Eloquence daemon is different, depending on operating system release. See below for further information.

The commands discussed in this section can only be executed by a “superuser”.

Starting the HP Eloquence daemon

When eloqd starts, the entries in the **`eloq.config`** file are read into a common memory region, shared by all HP Eloquence processes. This process can only be started once on a system.

The HP Eloquence daemon can be started manually by giving the following command:

```
/opt/eloquence/bin/eloqd
```

To check whether the eloqd program has been started, enter the following command:

```
ps -ef | grep eloqd
```

If it has been started, a display similar to the following appears:

```
root   2453     1  0  Feb 11  ?        0:00 /opt/eloquence/bin/eloqd
```

NOTE: It is recommended that the HP Eloquence daemon is started automatically on system boot.

Terminating the HP Eloquence daemon

The HP Eloquence daemon can be terminated manually by using the kill command. When the HP Eloquence daemon encounters the kill signal, it will terminate all currently running HP Eloquence processes and then terminate itself.

As an example, you may have in a script file:

```
pid=`ps -el | grep eloqd`
kill `echo $pid | cut -d" " -f4`
```

WARNING: **Never use a kill -9 to terminate the HP Eloquence daemon. This could lead to database consistency problems.**

Starting the HP Eloquence daemon automatically with HP-UX 9.x

In order to start the HP Eloquence daemon automatically on system boot, you should add the following command to your **`/etc/rc`** system startup script.

```
/opt/eloquence/bin/eloqd
```

Starting the HP Eloquence daemon automatically with HP-UX 10.x

The HP ELOQUENCE daemon eloqd is started automatically by default if run-level 2 is entered (after a reboot or single user mode) and will be shut down automatically on reboot.

The automatic eloqd start-up is configured in the start-up configuration file **`/etc/rc.config.d/eloquence`**.

To start eloqd on demand enter the following command:

```
/sbin/init.d/eloq start
```

To stop eloqd on demand enter the following command:

```
/sbin/init.d/eloq stop
```

Customize the ELOQ.INI File on the HP-UX System

ELOQ.INI is the HP Eloquence configuration file for the dialog server and related utilities. You can find it at the following location:

HP-UX 9.x	**/opt/eloquence/etc/eloq.ini**
HP-UX 10.x	**/etc/opt/eloquence/eloq.ini**

There is also a default HP-UX eloq.ini file at the location **/opt/eloquence/newconfig/config/eloq.ini**. If you update HP Eloquence, the eloq.ini is left untouched, so any changes required must be made manually.

HP-UX eloq.ini file format

The eloq.ini file contains several sections each containing a group of related configuration items. The sections and configuration items have the following format:

```
[Section]
Item=Value
```

Section is the name of a section. The enclosing brackets ([ ]) are required and they must start at the 1st column.

Item=Value defines a value of a configuration item. *Item* is the name of a configuration item. It consists of any sequence of characters (case insensitive) and digits followed by a equal sign (=). Depending on item type, the value may either be an integer or a string (optionally enclosed in double quotes).

Comment lines must start with a semicolon (;) or a hash character(#) in the first column.

User-defined sections

User-defined sections are used by the motif and alpha drivers.

These drivers are launched by the **`DLG SET ".driver"`** statement, the syntax of this statement is:

`DLG SET ".driver","driver_spec [ini_section [arguments]]"`

driver_spec: *motif* or *alpha*

ini_section: Optional name of an *user-defined section* in the eloq.ini file where the defaults specified in the section *[dmsrv]* can be overridden.

arguments: Additional arguments can optionally be specified here and will be passed-through to the Dialog Manager.

On driver start-up, the following tasks are performed:

1 The driver sets up the *Dialog Manager argument list* from the **`arguments`** specified in the **`DLG SET ".driver"`** statement, if any. The driver then reads the configuration items in the section *[dmsrv]* (please refer to the description of *Section [dmsrv]* below).

2 If **`ini_section`** is specified in the **`DLG SET ".driver"`** statement, the driver reads additional configuration items from this *user-defined section*. If these items have been previously specified in the section *[dmsrv]*, the previous values are overridden.

Example:

```
[debug]
DefaultsFile = /opt/eloquence/lib/debug/defaults.eq
```

This *user-defined section* named [debug] defines one item *DefaultsFile*. In order to activate this item, the name of this section must be specified in the **`DLG SET ".driver"`** statement, e.g.:

```
DLG SET ".driver","motif debug"
```

Any previous *DefaultsFile* definition in section *[dmsrv]* is overridden with the new value from section [debug].

3 If **`ini_section`** is specified in the **`DLG SET ".driver"`** statement, the driver searches this *user-defined section* for an item named *Arguments*. If this item exists, its value is appended to the *Dialog Manager argument list*.

Example:

```
[debug]
```

```
Arguments = -IDMtracefile /tmp/idmtrace
```

This enables an additional Dialog Manager argument which creates a trace file for debugging purposes.

4 Finally, the composed *Dialog Manager argument list* is passed to the Dialog Manager runtime system start-up function.

NOTE: If **`arguments`** are immediately specified in the **`DLG SET ".driver"`** statement, an *user-defined section* must be specified, too. However, if you specify a section name not present in the eloq.ini file, the *user-defined section* will be ignored.

NOTE: For a list of valid commandline arguments, please refer to the *ISA Dialog Manager* documentation.

Any configuration item valid for section *[dmsrv]* may also be defined in *user-defined sections* (please refer to the description of *Section [dmsrv]* below).

Additionally, *user-defined sections* may hold the following configuration item:

Arguments

Arguments=	<Additional Dialog Manager Arguments>
Default:	none
Function:	Additional arguments specific to the current *user-defined section* can optionally be specified here and will be passed-through to the Dialog Manager.

Section [dmsrv]

This section is used by the motif and alpha drivers.

The section [dmsrv] and any *user-defined sections* may hold the following configuration items:

DefaultsFile

DefaultsFile= <Filename>

Default: **`/opt/eloquence/lib/defaults.eq`**

If you are using modular dialog files, change this default to:

`/opt/eloquence/lib/module/defaults.eq`

Function: Defines the name (and path) of the defaults file used by motif and alpha drivers.

The defaults file is used to provide defaults to dynamically created objects.

IdmLib

IdmLib= <Search Path>

Default: none, the environment variable **`IDMLIB`** is used only.

Function: On execution of the **`DLG LOAD`** statement the specified dialog file is searched in each directory specified by the *IdmLib* item regardless of any directory specification named in the **`DLG LOAD`** statement. The directories are separated by '**`:`**'.

If *IdmLib* is specified in an *user-defined section*, its value is appended to the previous *IdmLib*, if any.

Example: **`IdmLib=/opt/eloquence/lib/app1:opt/eloquence/lib/app2`**

This causes every dialog file loaded by **`DLG LOAD`** to be searched in the directories **`/opt/eloquence/lib/app1`** and **`/opt/eloquence/lib/app2`**.

NOTE: The environment variable **`IDMLIB`** has the same purpose. If it is present, the contents of the *IdmLib* item is appended to the contents of the **`IDMLIB`** variable.

Dlg

Dlg= <Prefix>

Default: **`dlg`**

Function: This specifies the file extension, by which HP Eloquence recognizes a HP Eloquence dialog file.

Whenever HP Eloquence tries to load a Dlg file (for example sample.dlg), it will try to locate and load an Idc (e.g. sample.idc) or Idm (e.g. sample.idm) file first.

NOTE: Although it's possible to use HP Eloquence dialog files (they are converted internally at runtime) it's strongly recommended to convert them to Dialog Manager format due to performance considerations.

Idm

Idm= <Prefix>

Default: **`idm`**

Function: This specifies the file extension, by which HP Eloquence recognizes a Dialog Manager dialog file.

This file is usually created by the cvdlg utility program or the Dialog Manager graphical editor.

Idc

Idc= <Prefix>

Default: **`idc`**

Function: This specifies the file extension, by which HP Eloquence recognizes a compiled Dialog Manager dialog file.

This file is created by the idm utility program provided with the Dialog Manager product.

Debug

Debug= <Level>

Default: `0`

Function: DebugLevel specifies the driver debug level. A zero value is recommended. A nonzero value will cause debug messages output to stderr.

Items which control access to the on-line help

The Netscape WWW browser is now used, among other things, as an on-line help viewer. The following configuration items are necessary to establish this functionality.

Netscape

Netscape= <Path to Netscape>

Default: none

Function: The Netscape item is used by the dialog server as the location of the Netscape executable. This is vital in order to communicate with or to start-up Netscape.

If the 'Netscape' item is not present, the **`NETSCAPE`** environment variable is used instead. If this variable is also not present, the following access path will be assumed:

`/usr/bin/X11/netscape`

NOTE: This functionality requires Netscape revision 1.1 or above. Netscape is not included with HP Eloquence.

HelpBaseURL

HelpBaseURL= <URL>

Default: none

Function: This defines the base URL of the on-line help files.

Example: **`HelpBaseURL=http://www/help/`**

FileBaseURL

FileBaseURL= <URL>

Default: none

Function: This defines the base URL for the function **`EqHelpViewFile`**.

Example: **`FileBaseURL=http://www/files/`**

ManBaseURL

ManBaseURL= <URL>

Default: none

Function: This defines the base URL for the function **`EqHelpManPage`**.

Example: **`ManBaseURL=http://www/cgi-bin/man2html`**

Section [runsrv]

This section is used by HP Eloquence to communicate with the **RUNSRV** utility program on a remote host.

The section [runsrv] may hold the following configuration items:

Service

Service= <Servicename>

Default: **runsrv**

Function: Defines the service name (or port number) used to connect to **RUNSRV** on a remote host.

The service name is used to lookup the port number in your **/etc/services** file.

NOTE: You should provide the appropriate entry in your **/etc/services** file.

If the first character of the value is a digit, the value will be considered as a port number. It is recommended to use a service name.

The resulting port number *must* be the same on the client and server side.

Debug

Debug= <Level>

Default: **0**

Function: This is used internally to debug **RUNSRV** itself. The default value (zero) is recommended. A nonzero value will cause debug messages output to stderr.

Section [dmclnt]

This section is used by HP Eloquence to communicate with the dialog server on a remote host.

The section [dmclnt] may hold the following configuration items:

Service

Service= <Servicename>

Default: `dmclnt`

Function: Defines the service name (or port number) used by HP Eloquence to setup the connection to the dialog server on a remote host.

The service name is used to lookup the port number in your **`/etc/services`** file.

If the first character of the value is a digit, the value will be considered as a port number. It is recommended to use a service name.

The resulting port number *must* be the same on the client and server side.

NOTE: Depending on the *PortRange*, a range of port numbers is used, starting with a port number defined by the service dmclnt.

PortRange

PortRange= <Portnumber>

Default: `40`

Function: Defines the number of ports reserved for usage by the dialog server. This is also the upper limit of concurrent connected dialog servers.

If this number is nonzero, **`RUNSRV`** will be used to start the dialog server process, passing the actual port number.

Section [dmclnt@host]

To handle non-standard configurations, it's possible to overwrite default values as defined by the section [dmclnt]. To achieve this, define a section specific to a client PC by appending the PC host name to dmclnt separated by an at sign (@).

The section [dmclnt@host] may hold the following configuration items:

Service

Service= <Servicename>

Default: value taken from [dmclnt] section

Function: Defines the service name (or port number) used by HP Eloquence to setup the connection to the dialog server on a remote hosts.

The service name is used to lookup the port number in your **`/etc/services`** file.

If the first character of the value is a digit, the value will be considered as a port number.

This provides a nonstandard way to connect to the client PC. It is recommended to use the default service name.

PortRange

PortRange= <Portnumber>

Default: value taken from [dmclnt] section

Function: Defines number of ports reserved for usage by dialog server.

If the PortRange is zero, only a single port number (as defined by *Service*) will be used to connect the dialog server.

This provides a nonstandard way to connect to the client PC.

Setup your Network Connection on the PC

To setup your network connection you change the following network software configuration files:

SERVICES	maps service names and port numbers
HOSTS	maps IP addresses and host names

The location of these files depend on your network software. Typical locations are:

- c:\lanman\etc
- c:\wrqnet
- c:\windows

Change your HOSTS file

Add the name of your HP 9000 server to your **HOSTS** file. Make an entry like below:

```
192.1.1.1    hostname
```

Where *192.1.1.1* is the IP address of your HP9000 system and *hostname* is the hostname of your HP9000 system.

Change your SERVICES file

To provide a service name / port number mapping, you change your **SERVICES** file.

Add lines like below to your **SERVICES** file:

```
eloq      8000/tcp    # HP Eloquence remote database
runsrv    8010/tcp    # HP Eloquence RUNSRV utility
```

Where

eloq is the service name used by HP Eloquence remote database access.

runsrv is the service name used by the HP Eloquence **RUNSRV** utility.

Verify carefully that the chosen port numbers (8000,8010 in the example above) are unique and not already in use.

Customize the eloq.ini file

The file **ELOQ.INI** must be located in your **WINDOWS** directory. By changing **ELOQ.INI** it's possible to customize **RUNSRV** and **DLGSRV**. Please refer to the section *Customize the eloq.ini file on the PC platform* later in this chapter for further information.

No changes are necessary to the **ELOQ.INI** file, unless the defaults are not sufficient. All paths are updated accordingly by the setup program.

Customize the ELOQ.INI File on the PC Platform

The ELOQ.INI file must be installed in your **`WINDOWS`** directory. You may change it using a text editor (such as MS DOS edit or Microsoft Windows Notepad).

If you update HP Eloquence, your ELOQ.INI file is renamed to ELOQ.BAK and a new ELOQ.INI file is installed.

File format

The file ELOQ.INI contains several sections each containing a group of related configuration items. The sections and configuration items have the following format:

```
[Section]
Item=Value
```

Section is the name of a section. The enclosing brackets ([ ]) are required and they must start at the first column.

Item=Value defines a value of a configuration item. *Item* is the name of a configuration item. It consists of any sequence of characters (case insensitive) and digits followed by a equal sign (=). Depending on item type, the value may either be an integer or a string (optionally enclosed in double quotes).

Comment lines must start with a semicolon (;) in the first column.

Section [runsrv]

This section is used by the **RUNSRV** utility program. It may hold the following configuration items:

Service

Service= <Servicename>

Default: **runsrv**

Function: Defines the service name (or port number) **RUNSRV** expects to become connected from a remote host (running HP Eloquence).

The service name is used to lookup the port number in your **SERVICES** file.

If the first character of the value is a digit, the value will be considered as a port number. It is recommended to use a service name.

NOTE: You should provide the appropriate entry in your **SERVICES** file. The location of your **SERVICES** file depends on your networking software.

Debug

Debug= <level>

Default: **0**

Function: DebugLevel specifies the **RUNSRV** debug level. A zero value is recommended.

DlgSrvX

DlgSrvX= <Command to start **DLGSRV**>

Default: **DLGSRV%s -connect %s -IDMfont 1 -IDMcolor 1**

Function: This specifies the command line that is executed by **RUNSRV** to launch **DLGSRV**. It must contain two entries of **%s**.

If an instance of **DLGSRV** is already running, the first entry of **%s** will be replaced by the **DLGSRV** instance appendix. This is necessary since Microsoft Windows denies to run more than one instance of **DLGSRV** at the same time (please refer to the *HP*

Eloquence Graphical User Interface documentation for details).

The second entry of **%s** will be replaced by the connection argument provided by HP Eloquence (please refer to the **DLG-SRV** documentation for details).

NOTE: The *NDlgSrv* item must be setup properly if multiple instances of **DLGSRV** are to be run at the same time (see below).

NDlgSrv

NDlgSrv= <Number of **DLGSRV** Instances>

Default: **1**

Function: This specifies the maximum number of **DLGSRV** instances to be run at the same time.

After changing the *NDlgSrv* value, you should run the **DLG-CLONE** utility program which automatically creates the required number of **DLGSRV** clone-copies (please refer to the *HP Eloquence Graphical User Interface* documentation for details).

NOTE: The *DlgSrvX* item must be present if multiple instances of **DLGSRV** are to be run at the same time (see above).

DlgSrv

DlgSrv= <Command to start **DLGSRV**>

Default: **DLGSRV -connect %s -IDMfont 1 -IDMcolor 1**

Function: This specifies the commandline that is executed by **RUNSRV** to launch **DLGSRV**.

An entry of **%s** will be replaced by the connection argument provided by HP Eloquence (please refer to the **DLGSRV** documentation for details).

NOTE: *DlgSrv* serves for backward compatibility and may not be supported in future releases anymore. *DlgSrvX* is much more flexible and should be used instead.

User-defined sections

User-defined sections are used by the **DLGSRV** utility program.

The **DLGSRV** utility program is launched by the **DLG SET ".driver"** statement, the syntax of this statement is:

DLG SET ".driver","driver_spec [ini_section [arguments]]"

driver_spec: @*hostname*

hostname is the name of the system where the **DLGSRV** utility program shall be executed.

ini_section: Optional name of an *user-defined section* in the eloq.ini file where the defaults specified in the section *[dlgsrv]* can be overridden.

arguments: Additional arguments can optionally be specified here and will be passed-through to the Dialog Manager.

On **DLGSRV** start-up, the following tasks are performed:

1 **DLGSRV** sets up the *Dialog Manager argument list* from the **arguments** specified in the **DLG SET ".driver"** statement, if any. **DLGSRV** then reads the configuration items in the section *[dlgsrv]* (please refer to the description of *Section [dlgsrv]* below).

2 If **ini_section** is specified in the **DLG SET ".driver"** statement, **DLGSRV** reads additional configuration items from this *user-defined section*. If these items have been previously specified in the section *[dlgsrv]*, the previous values are overridden.

Example:

```
[debug]
DefaultsFile = C:\DLG\DEBUG\DEFAULTS.EQ
```

This *user-defined section* named [debug] defines one item *DefaultsFile*. In order to activate this item, the name of this section must be specified in the **DLG SET ".driver"** statement, e.g.:

```
DLG SET ".driver","@my-pc debug"
```

Any previous *DefaultsFile* definition in section *[dlgsrv]* is overridden with the new value from section [debug].

3 If **ini_section** is specified in the **DLG SET ".driver"** statement, **DLGSRV** searches this *user-defined section* for an item named *Arguments*. If this item exists, its value is appended to the *Dialog Manager argument list*.

Example:

```
[debug]
Arguments = -IDMtracefile C:\TMP\IDMTRACE.TXT
```

This enables an additional Dialog Manager argument which creates a trace file for debugging purposes.

4 Finally, the composed *Dialog Manager argument list* is passed to the Dialog Manager runtime system start-up function.

NOTE: If **`arguments`** are immediately specified in the **`DLG SET ".driver"`** statement, an *user-defined section* must be specified, too. However, if you specify a section name not present in the eloq.ini file, the *user-defined section* will be ignored.

NOTE: For a list of valid commandline arguments, please refer to the *ISA Dialog Manager* documentation.

Any configuration item valid for section *[dlgsrv]* may also be defined in *user-defined sections* (please refer to the description of *Section [dlgsrv]* below).

Additionally, *user-defined sections* may hold the following configuration item:

Arguments

Arguments=	<Additional Dialog Manager Arguments>
Default:	none
Function:	Additional arguments specific to the current *user-defined section* can optionally be specified here and will be passed-through to the Dialog Manager.

Section [dlgsrv]

This section is used by the **DLGSRV** utility program.

The section [dlgsrv] and any *user-defined sections* may hold the following configuration items:

MemSz

MemSz= <MemorySize>

Default: **4096**

Function: MemorySize specifies the **DLGSRV** communication memory size. The arguments exchanged by server and client is limited by the communication memory. A value of 4096 is recommended.

DefaultsFile

DefaultsFile= <Filename>

Default: **C:\DLG\DEFAULTS.EQ**

If you are using modular dialog files, change this default to:

C:\DLG\MODULE\DEFAULTS.EQ

Function: Defines the name (and path) of the defaults file used by the **DLGSRV** program. If undefined, **DLGSRV** will look for a file defaults.eq in the current directory.

The defaults file is used to provide defaults to dynamically created objects.

IdmLib

IdmLib= <Search Path>

Default: none, the environment variable **IDMLIB** is used only.

Function: On execution of the **DLG LOAD** statement the specified dialog file is searched in each directory specified by the *IdmLib* item regardless of any directory specification named in the **DLG LOAD** statement. The directories are separated by '**;**'.

If *IdmLib* is specified in an *user-defined section*, its value is appended to the previous *IdmLib*, if any.

Example: `IdmLib=C:\DLG\APP1;C:\DLG\APP2`

This causes every dialog file loaded by `DLG LOAD` to be searched in the directories `C:\DLG\APP1` and `C:\DLG\APP2`.

NOTE: The environment variable `IDMLIB` has the same purpose. If it is present, the contents of the *IdmLib* item is appended to the contents of the `IDMLIB` variable.

DlgPath

DlgPath= <Path>

Default: none

Function: Defines the path where `DLGSRV` expects dialog files.

NOTE: If *IdmLib* and/or the environment variable `IDMLIB` is present, *DlgPath* will be ignored. *DlgPath* serves for backward compatibility and may not be supported in future releases anymore. *IdmLib* is much more flexible and should be used instead.

Dlg

Dlg= <Prefix>

Default: `DLG`

Function: This specifies the file extension, by which HP Eloquence recognizes a HP Eloquence dialog file.

Whenever HP Eloquence tries to load a Dlg file (for example sample.dlg), it will try to locate and load an Idc (e.g. sample.idc) or Idm (e.g. sample.idm) file first.

NOTE: Although it's possible to use HP Eloquence dialog files (they are converted internally at runtime) it's strongly recommended to convert them to Dialog Manager format due to performance considerations.

Idm

Idm= <Prefix>

Default: `DLG`

Function: This specifies the file extension, by which HP Eloquence recognizes a Dialog Manager dialog file.

This file is usually created by the cvdlg utility program or the Dialog Manager graphical editor.

Idc

Idc= <Prefix>

Default: `IDC`

Function: This specifies the file extension, by which HP Eloquence recognizes a compiled Dialog Manager dialog file.

This file is created by the idm utility program provided with the Dialog Manager product.

Debug

Debug= <Level>

Default: `0`

Function: DebugLevel specifies the **`DLGSRV`** debug level. A zero value is recommended.

Logfile

Logfile= <Filename>

Default: none

Function: Defines the name (and path) of a log file **`DLGSRV`** uses to write debug messages to. An empty value (default) causes **`DLGSRV`** to write to dbwin (using **`OutputDebugString`**). This value is only used if DebugLevel is nonzero.

Host

Host= <Hostname>

Default: none

Function: Defines the host name **`DLGSRV`** will use to connect to the remote host (running HP Eloquence).

NOTE: This configuration item provides a nonstandard way connecting to a HP Eloquence process. Using this option is not recommended.

Service

Service= <Servicename>

Default: none

Function: Defines the service name (or port number) **DLGSRV** will use to connect to the remote host (running HP Eloquence).

The service name is used to lookup the port number in your **SERVICES** file.

If the first character of the value is a digit, the value will be considered as a port number. It is recommended to use a service name.

NOTE: You should provide the appropriate entry in your **SERVICES** file. The location of your **SERVICES** file depends on your networking software.

NOTE: This configuration item provides a nonstandard way connecting to a HP Eloquence process. Using this option is not recommended.

Items which control access to the on-line help

The Netscape WWW browser is now used, among other things, as an on-line help viewer. The following configuration items are necessary to establish this functionality.

NOTE: Since this functionality uses the Microsoft Windows DDE protocol, you should provide a *Netscape* item in the section *[modules]*. If this item is missing, **DLGSRV** will not be able to automatically start-up Netscape. Please refer to the description of *Section [modules]* below and to the *HP Eloquence Graphical User Interface* documentation for details.

NOTE: This functionality requires Netscape revision 1.1 or above. Netscape is not included with HP Eloquence.

HelpBaseURL

HelpBaseURL= <URL>

Default: none

Function: This defines the base URL of the on-line help files.

Example: **`HelpBaseURL=http://www/help/`**

FileBaseURL

FileBaseURL= <URL>

Default: none

Function: This defines the base URL for the function **`EqHelpViewFile`**.

Example: **`FileBaseURL=http://www/files/`**

ManBaseURL

ManBaseURL= <URL>

Default: none

Function: This defines the base URL for the function **`EqHelpManPage`**.

Example: **`ManBaseURL=http://www/cgi-bin/man2html`**

Section [modules]

This section is used by the **RUNSRV** and **DLGSRV** utility programs in context with the Microsoft Windows DDE communication protocol.

Since DDE server programs are not automatically started by Microsoft Windows, this section associates the Microsoft Windows *module identifier* with the program execution path. This enables the **RUNSRV** and **DLGSRV** utility programs to start-up the required DDE server programs themselves if they are not currently running.

If you want to communicate with any Microsoft Windows program using DDE, knowledge about the Microsoft Windows *module identifier* and the communication topics specific to the program is required. Normally, these informations are contained in the program's documentation.

Example:

```
[modules]
WinWord = C:\WINWORD\WINWORD.EXE
```

This associates the *module identifier* of *Microsoft Word for Windows* with the program execution path. This enables the **RUNSRV** and **DLGSRV** utility programs to establish DDE communications with *Microsoft Word for Windows*.

The **DLGSRV** utility program has built-in support for DDE communication with the Netscape WWW browser. This is used to establish access to the on-line help system. In order to use this, you do not need any information concerning the internal Netscape DDE implementation. All you need to do is to add an entry in the section [modules] so that Netscape can be automatically started.

Example:

```
[modules]
Netscape = C:\NETSCAPE\NETSCAPE.EXE
```

NOTE: The Microsoft Windows *Task List* displays the *module identifiers* of all programs currently running.

NOTE: Please refer to the *HP Eloquence Graphical User Interface* documentation for details about *On-line Help* and *RUNSRV DDE Communication*.

3

HP Eloquence A.05.10 language improvements

This chapter covers the new language functionality implemented in HP Eloquence A.05.10.

XPACK, XUNPACK statements

The XPACK and XUNPACK statements provide a convenient way to transfer string and numeric data to and from a string variable. They are particular useful in conjunction with Dialog Manager record objects.

Unlike the PACK USING and UNPACK USING statements, the XPACK and XUNPACK statements include the variable name in the packed string and use a variable length string format.

The XPACK statement

The XPACK statement transfers data from each variable of the variable list to the destination string. A variable list may be specified in three different ways:

- a string expression evaluating in a variable list
- a variable list
- a reference to IN DATA SET LIST variable lists

A reference to a whole array will be resolved into a list of array elements.

As the data is transferred to the destination string, it is converted into string format and stored along with the variable name and array index.

XPACK ... USING

XPACK Dest$ USING <string expression>

Pack destination string variable from variable list specified by the string expression. The string expression contains a list of variable names separated by a comma.

For example:

```
List$="A,A$,B,B$,A$(1),Array$(*)"
```

XPACK ... USING REMOTE LISTS

XPACK Dest$ USING REMOTE LISTS Label [,Label ...]

Pack destination string variable from variable list as defined by the referenced remote list(s).

For example:

```
XPACK Dest$ USING REMOTE LISTS Label1,Label2
Label1: IN DATA SET LIST A,A$
Label2: IN DATA SET LIST ...
```

XPACK ... FROM

`XPACK Dest$ FROM Variable [,Variable ...]`

Pack destination string variable from variable list.

For example:

```
XPACK Dest$ FROM A,A$,B,B$,A$(*)
```

The XUNPACK statement

The XUNPACK statement transfers data from a string variable into the original variables. If a variable list is specified, only those variables are unpacked, that are included in the variable list. It's not possible to unpack a buffer into a different variable than used to pack the buffer.

`XUNPACK Buf$`

Unpack buffer variable into variables as named in the buffer.

For example:

```
A$="Test"
B=123
XPACK Buf$ FROM A$,B
A$=""
B=0
XUNPACK Buf$
```

This recovers the initial values of A$ and B.

`XUNPACK Buf$ USING <string expression>`

Unpack buffer variable into variables as named in the buffer. The string expression contains a list of variable names separated by a comma. Only variables included in the variable list are unpacked.

For example:

```
XPACK Buf$ FROM A$,B,C
List$="A$,B"
XUNPACK Buf$ USING List$
```

This should unpack the variables A$ and B only.

`XUNPACK Buf$ FROM Variable [,Variable ...]`

Unpack buffer variable into variables as named in the buffer. Only variables included in the variable list are unpacked.

For example:

```
XPACK Buf$ FROM A$,B,C
XUNPACK Buf$ FROM A$,B
```

This unpacks the variables A$ and B only.

XUNPACK Buf$ USING REMOTE LISTS Line_id[,Line_id ...]

Unpack buffer variable into variables as named in the buffer. Only variables included in the referenced REMOTE LISTS list are unpacked.

For example:

```
XPACK Buf$ FROM A$,B,C
XUNPACK Buf$ USING REMOTE LISTS Label
Label: IN DATA SET LIST A$,B
```

XPACK format description

Each variable in a XPACK statement is converted into a string format containing fields describing variable type, name, index and value.

The following rules apply:

- Format fields are separated by tilde characters ('~').
- All fields are printable (no binary data).
- Numeric values are converted into string.

Table 1 **Format Fields**

Type	Name	~	Idx	~	Len	~	Value	...	$

~ Field separator. The Tilde character ('~').

Type Field type. One of the following:

N numeric
X string
$ End-of-list

Name Name = field name. Up to 15 characters. Must be valid Variable name. Trailing $ of string variables is omitted.

Idx Array index. 0 = simple variable.

First array element is 1 (independent of array bounds and number of dimensions).

Len Size (number of bytes) of value field.

Value Value field. Any number of characters.

Simple numeric variable

```
Simple = 47
XPACK Buf$ FROM Simple
```

Results in the following buffer:

```
   "NSimple~0~2~47$"
```

Equivalent Eloquence code:

```
Buf$="N"&"Simple~"&"0~"&VAL$(LEN(VAL$(Simple)))
     &"~"&VAL$(Simple)&"$"
```

Simple string variable

```
Simple$ = "String"
XPACK Buf$ FROM Simple$
```

Results in the following buffer:

```
   "XSimple~0~6~String$"
```

Equivalent Eloquence code:

```
Buf$="X"&"Simple~"&"0~"&VAL$(LEN(Simple$))&"~"&Simple$&"$"
```

Array element

```
A$(1)="TEST"
XPACK Buf$ USING "A$(1)"
```

Results in the following buffer:

```
   "XA~1~4~TEST$"
```

Array

```
DIM A$(0:3)
A$(0)="000"
A$(1)="111"
A$(2)="222"
A$(3)="333"
XPACK Buf$ FROM A$(*)
```

Results in the following buffer:

```
"XA~1~3~000XA~2~3~111XA~3~3~222XA~4~3~333$"
```

Example Eloquence XUNPACK program

The example program below shows how to unpack a buffer in XPACK format. This example has been provided only for clarification of XPACK format.

```
DIM Type$[1],Var_name$[15],Value$[80],Variable$[22]
INTEGER P,Idx,Len

LOOP
   Type$=Buf$[1,1]             ! Type
   EXIT IF Type$="$"
   Buf$=Buf$[2]
```

```
   P=POS(Buf$,"~")              ! locate separator
   Var_name$=Buf$[1;P-1]        ! Variable name
   Buf$=Buf$[P+1]

   P=POS(Buf$,"~")              ! locate separator
   Idx=VAL$(Buf$[1,P-1])        ! Array index
   Buf$=Buf$[P+1]

   P=POS(Buf$[P],"~")           ! Separator
   Len=VAL(Buf$[1,P-1])         ! Value length
   Buf$=Buf$[P+1]

   Value$=Buf$[1;Len]           ! Value
   Buf$=Buf$[Len+1]

   IF Idx THEN
      Variable$=Var_name$&"("&VAL$(Idx)&")"
   ELSE
      Variable$=Var_name$
   END IF

   IF Type$="X" THEN
      COMMAND Variable$&"$=Value$"
   ELSE
      COMMAND Variable$&"="&Value$
   END IF
END LOOP
```

COMMAND statement enhancements

The syntax for the **COMMAND "!…"** statement has been extended. It now supports the specification of a result string. If present, the output of a HP-UX command will be returned in the result string instead being output to the **SYSTEM PRINTER**.

For example:

```
COMMAND "!uname -i",Serial$
DISP "Your serial number is ";Serial$
```

Example above displays your system serial number (as returned by the uname -i command).

HOP key

In addition of entering the HOP statement in order to execute a 'step over' or 'advanced step' operation during program debugging, you may now press ^G (ctrl-G).

LASTLINE

The LASTLINE keyword does return the last line number used in current program.

This makes it more convenient, to dynamically load subprograms or functions below the end of the program.

For example:

```
LOAD SUB "SUBX",LASTLINE+1,1
```

will load the subprograms and functions from program file SUBX renumbering to the next available line number.

New ASSIGN mode

The syntax for the **ASSIGN** and **XASSIGN** statements have been extended. Now an **APPEND** file mode is supported. It will cause all output to a HP-UX sequential file automatically appended to the end of the file.

If a file is opened in **APPEND** mode, it will behave different:

- If file does not exist, it will be created upon ASSIGN with a zero size.
- File is *not* locked exclusive. Because all output will automatically appended to the end of the file, it's not necessary to protect file contents by locking it. If you need exclusive access, you may still use the **LOCK #** and **UNLOCK #** statements.

For example:

```
ASSIGN #1 TO "Logfile";APPEND
PRINT #1;DATE$;TIME$;"Message"
...
```

The example code above will cause output appended to Logfile. If Logfile does not exist, it will be created.

Maximum number of open files

The maximum number of files, that may be ASSIGNed at one time has been extended from 10 to 20.

When the limit of maximum number of open files is exceeded, a runtime error 1000 (“System files table full or maximum number of files already assigned”) is returned by the ASSIGN statement.

Please note that you still can ASSIGN only 10 files per segment.

FNAME$

The keyword

```
FNAME$( expr )
```

returns the file name associated with the given file number.

For example:

```
ASSIGN #1 TO "foo,TMP"
DISP FNAME$(1)
```

-> /tmp/foo.DATA

An absolute HP-UX compliant path/name is returned.

store utility enhancements

The HP Eloquence store utility is a HP-UX command line tool to create HP Eloquence programs from text files. It is only available for the HP Eloquence development environment.

The store utility has been enhanced:

- store is now able to deal with tab characters in the input text
- store supports directives
- store includes the name of the input file in a syntax error message.
- If the -e command line argument is present, store will output its error messages in a different form
- store does now understand the line and file information provided by the C preprocessor

Synopsis:

```
Usage: store [options] inputfile
options:
 -help   = Show usage (this list)
 -o file = specify output filename. Default is 'inputfile.PROG'
           (Note this MUST be specified if reading from stdin)
 -n      = don't create output file
 -v      = Verbose. Output source lines
 -s      = Use space dependent mode
 -t wdth = tab width (default is 8, 0 = off)
 -f lno  = first line number (default is 1)
 -i inc  = line number increment (default is 1)
 -e      = change error message form
Specifying '-' as inputfile will force reading from stdin
```

New command line options:

-t wdth tab width (default is 8, 0 = off)

When editing your programs in an HP-UX editor (like vi) rather than the integrated HP Eloquence editor, tab characters provide a convenient way to format your source code.

Tab characters in the source code caused a syntax error. When specifying a tab width, all tab characters read from input file are replaced by the appropriate number of spaces. A zero tab width will disable tab expansion.

This may also be controlled by the $TAB directive.

-f lno first line number (default is 1)

When no line numbers are included in source code, store will provide its own, starting with 1.

This commandline switch makes it possible to define the first line number to use for automatically generated line numbers.

This may also be controlled by the $LINE directive.

-i inc line number increment (default is 1)

This commandline switch makes it possible to define the increment, used for automatically generated line numbers.

This may also be controlled by the $LINE directive.

-e If the -e commandline argument is present, store will output error messages is a format more suitable for automatic processing.

If the -e commandline argument is present, the error message will be of the following format:

```
store: "file.in", line 1: IMPROPER STATEMENT
^      ^          ^       ^
|      |          |       The error message
|      |          The line number of the input file
|      The name of the input file or stdin if read from stdin
To indicate it's a message from the store utility
```

For example:

Put the invalid statement below in a file named "file.in":

```
STOP HERE
```

Without the -e commandline argument, store will output the following error message:

```
STOP HERE
     ^
IMPROPER STATEMENT
store: "file.in", line 37
```

With the -e commandline argument, store will output the following error message:

```
store: "file.in", line 1: IMPROPER STATEMENT
```

Directives

The following directives, if included in the source code, make it possible to control the behavior of some store options within the code.

```
$TAB wdth
```

This will define a different tab character handling.

For example:

```
$TAB 8
```

Sets the tab expansion to 8 for subsequent source lines.

```
$LINE fline,inc
```

Defines a different automatically generated line numbers.

For example:

```
$LINE 1000,10
```

Presets the next automatically generated line number to 1000 and the line increment to 10.

```
$LINE 1000
```

Presets the next automatically generated line number to 1000 and the line increment to 10.

Using store with the C preprocessor

The C preprocessor is a macro processor which provides the following functionality:

- include files
- define and replace macros

Using the C preprocessor with Eloquence makes it possible to use the preprocessor functionality.

For example:

```
! This is a sample program HP Eloquence program using the
! C preprocessor

#define MAXLOOP 10

SUB Sample(INTEGER Partno)

! include the common block
#include "common"

! The subprogram body
   DBASE IS Db$
   IN DATA SET "PARTS" USE REMOTE LISTS Part_1
   DBGET(Db$,"PARTS",7,S(*),"@",Buf$,Partno)
   ...
   FOR I=1 TO MAXLOOP
```

```
      ...
   NEXT I
   ...
! include the definitions for PARTS
#include "PARTS"

SUBEND
```

To make this a HP Eloquence PROG file, you could use the following command line:

```
cc -E sample | store -o SAMPLE -
```

The C preprocessor replaces the #include directives by the referenced file content and the macros by the definitions. When a syntax error in an included file is detected, store is now able to understand the preprocessor location information and will report the file name and position.

Please refer to the appropriate HP-UX documentation for more information.

Program file size limit

HP Eloquence program files are no longer limited to a file size of 64k.

Due to HP260 program format limitations, an HP Eloquence program files were limited to a total size of 64 kbytes.

An error 63 (“Number of records or bytes per record exceeds 999999 or program too big”) was returned, if you tried to store a program, which exceeded the 64 kbytes file limit.

If a program requires more than 64 kbytes to store, a different program file format is used. Programs below 64 kbytes will continue to use the former program file format to retain compatibility with former HP Eloquence releases.

NOTE: A program with a file size above 64 kbytes will not be compatible with HP Eloquence releases before A.05.02.

HP Eloquence keyboard handling

Two new statement groups allow better control of the Eloquence keyboard processing.

Typeahead

HP Eloquence normally rejects any key pressed while it is busy with a beep signal. The TYPEAHEAD statement enables the program to control HP Eloquence keyboard handling

```
TYPEAHEAD n
```

The typeahead statement adjusts the keyboard mode. Changing the keyboard mode implies a TYPEHEAD CLEAR operation.

n = 0 Default behavior. Any key pressed while HP Eloquence is not in input state will be rejected.

n = 1 Partial typeahead. Any key, which is not a function key is saved in a typeahead buffer for later processing.

Function keys are executed immediately.

n = 2 Full typeahead. All keys are saved in the typeahead buffer for later processing.

In TYPEAHEAD mode 1, the program must be aware, that there may be some saved keystrokes, if reacting upon a function key.

In TYPEAHEAD mode 2, it's not possible to interrupt a program (for example abort a very long printout) using a function key. Function keys loose their meaning, while not in INPUT mode.

TYPEAHEAD CLEAR

Clear the typeahead buffer. If a character is currently in typeahead buffer, a beep is output to indicate the loss of input.

```
N=TYPEAHEAD
```

The typeahead function returns the number of characters currently in typeahead buffer.

ON KEYBD

The ON KEYBD statement makes it possible to use arbitrary keys like functions keys.

Syntax:

ON KEYBD #n[,#n...][,priority] {GOTO|GOSUB|CALL} target

NOTE: Key numbers are defined by curses and returned by the KBCODE keyword. You can't catch the break key (key number 0).
There is a maximum of 32 active ON KEYBD in one program segment
An ON KEYBD statement overrides any default sense of the key. Catching an important key (like carriage return) may result in a unusable keyboard within HP Eloquence

NOTE: A lot of special keys makes it hard to remember for the user, which keys are active or what keys result in what function. It's most likely a bad idea to catch any regular characters or control characters. If you catch some special keys (for example next line), the program reaction should be closely implied by the key.

OFF KEYBD

The OFF KEYBD# removes the interrupt handling for the specified key or all keys, if no key number has been specified.

Syntax:

```
OFF KEYBD #[n[,n...]]
```

LASTKEY

The LASTKEY keyword returns the key number of the last key that caused an ON KEYBD interrupt.

CURKEY

The CURKEY keyword returns 55 if the last interrupt was caused by an ON KEYBD statement.

Alternate definition of the Backspace key

`kbs` terminfo definition will be now recognized as an alternate backspace key. If **`kbs`** is set to DEL character, a DEL character will no longer cause screen refresh (you’ll have to use the ^L key).

132 columns on 700/60 terminal

132 column mode is now supported on 700/60 terminals too.

4

HP Eloquence Database

This chapter describes the added database functions since HP Eloquence version A.04.00.

The DBUTIL program

Introduction

The new DBUTIL database maintenance utility is a replacement for the former DBMODS utility (which is still available as of HP Eloquence release A.05.10, but will be removed in the next release).

Dbutil provides all functionality of the former DBMODS utility and additionally provides the ability to change some aspects of the database structure without having to export/import the database.

Database changes can either be defined interactive or by a script file (called control script). This makes it possible for a software vendor to provide a control script to a customer to perform database changes without manual interaction. The dbutil may even be used to interactively create a control script.

The following actions can be performed:

- ❒ Set the maintenance password
- ❒ Define access classes
- ❒ Grant or revoke access to datasets
- ❒ Create a new data item
- ❒ Change data item definition
- ❒ Create a new index item
- ❒ Change index item definition
- ❒ Create a new data set
- ❒ Add new data items to a data set
- ❒ Add new indices to a data set
- ❒ Additionally, in interactive mode, it is possible to print a schema definition of the database.

DBUTIL commandline arguments

Synopsis:

```
usage: dbutil [options] [file]

options:

 -help   = show usage (this list)
 -i      = interactive mode (batch mode only)
 -n      = pretend          (batch mode only)
 -v      = verbose          (batch mode only)
 -e cnt  = abort processing after encountering cnt errors
 -t tmp  = where temporary files are stored
```

```
  -d       = debug
```

If a file argument is present, dbutil will process in batch mode unless the **-i** option is present. If the **-n** option is present, no changes will be made to the database. Processing will end after checking the input file and the analysis phase.

Arguments:

-i After processing the supplied batch file, dbutil will change into interactive mode, so you can verify the database changes made so far or you can add additional changes. Interactive mode is the default, if no control file is specified on the command line.

-n dbutil will analyze the control file but will not make any changes to the database. This option can be used to check the syntax of the control file or to get an overview which changes would be applied to the database.

-v[v] Specifying the -v option will cause dbutil to output a summary of changes after analyzing the control file and some descriptive text during the database restructuring.

Specifying two -v options will cause dbutil to echo the control file to stdout as it is analyzed.

-e cnt Abort processing the control file after encountering the given number of syntax or validation errors.

-t tmp This option makes it possible to specify where temporary files will be created. As temporary files can become quite huge, it may overflow the default location.

If the -t argument is not specified, dbutil will allocate temporary files at the same location as the database.

-d This is used internally to debug dbutil itself. You should not use it.

If a file argument is present, dbutil will process in batch mode unless the **-i** option is present.

Dbutil execution

Dbutil will execute in different phases:

1. If a control file is given on the command line, it will analyze the control file and all changes imposed by the control file.
2. If the interactive mode is requested by either specifying the -i argument or by omitting

the control file, it will change into interactive mode.

3 Finally it will analyze all changes and list the restructure operations it needs to perform. If the -n argument is present execution will stop here.

4 If stdout and stdin are connected to a tty device, dbutil will prompt for a confirmation to apply the database changes. This is the last chance to change your mind. If you interrupt the program beyond this point, your database will be lost and you have to restore it from your backup.

5 Finally dbutil will start to restructure the database.

Example

```
$ dbutil test/du5

B1368B DBUTIL (C) COPYRIGHT HEWLETT-PACKARD GmbH 1995 (A.05.10)

Analyzing restructure specification ...
Analysis completed successfully

Checking database consistency ...
Consistency check completed successfully

Database restructure analysis:

Analyzing changes:
 NEW-SET-M
 * this is a new data set
   Estimated temporary disc space required: 1 MB

 NEW-SET-D
  * this is a new data set
  Estimated temporary disc space required: 1 MB

Estimated temporary disc space required: 1 MB

Temporary files will be created at
 /disc/project/eloq/db2/ndbmods/test/db/

Available disc space: 147 MB

Data restructure process required.


------------------------------------------------------------
PLEASE NOTE:
If the restructure process fails or is interrupted, YOUR
DATA BASE IS LOST and can only be recovered from a backup.
If you don't have a current backup, you should NOT continue.
------------------------------------------------------------

Continue with database restructure (y/n) ? y

CAUTION: DO NOT INTERRUPT THIS PROGRAM!

Removing current ROOT file

Restructuring database ...
NEW-SET-M
NEW-SET-D
```

```
Dumping new ROOT file /disc/project/eloq/db2/ndbmods/test/db/DB
End of Database Maintenance Utility
```

Control file syntax

The dbutil control file is a plain text file. The following general rules apply:

- Everything after a hash character (#) is considered a comment and will be ignored.
- dbutil does recognize keywords in either upper or lower case (but not mixed).
- Each statement must be delimited by a semicolon (;)
- Identifiers such as Item, Index and Set-names may be specified in either case (upper/lower case).
- Strings must be enclosed in double quotes. To include a quote character in a string, the quote character must be preceded by a backslash (\) character.
- A statement may span multiple lines.
- Some statements allow repeating a list of definitions by enclosing it in curly braces.

The syntax description below use the following conventions:

- All keywords are given in upper case.
- All Identifiers, such as Item, Index and Set names are given in upper/lower case.
- Optional syntax elements are enclosed in brackets.

Opening the database

`OPEN DATABASE "database" [PASSWORD "password"] ;`

database The database argument specifies the database path and name. The database path may be specified either by giving a HP-UX path or a HP Eloquence volume.

password The password argument specifies the database maintenance password. The PASSWORD clause is not required if the database has no maintenance password or you are the superuser (root).

For example:

```
OPEN DATABASE "/opt/sqlr/db.g/db";
```

This opens the database "db" located in directory **`/opt/sqlr/db.g`**.

```
OPEN DATABASE "test,DB" PASSWORD "secret";
```

This opens the database "TEST" at the path described by the HP Eloquence volume "DB".

Changing the maintenance password

```
ALTER MAINTENANCE PASSWORD TO "password";
```

This will set the database maintenance password as specified by the string argument. The maintenance password may not exceed 16 characters.

Changing access passwords

The HP Eloquence database has 32 access profiles. The access profile is selected during database open, depending on the given password. If no password is given, or there is no password at all, access profile 0 (PUBLIC) is selected.

ALTER PASSWORD
  acl_no ["password"];

```
ALTER PASSWORD {
   acl_no ["password"];
   ...
}
```

Access profiles 1 to 31 are associated with a password. Omitting the password (and specifying only the access class) will cause the password for this class to be removed and the access class become disabled. The password must not exceed 8 characters.

If enclosed in curly braces, multiple passwords can be defined.

For example:

```
CHANGE PASSWORD
   1 "GuessMe";
```

This defines the password "GuessMe" for access profile 1.

```
CHANGE PASSWORD {
    3;
    4 "Manager";
}
```

This statement disables the access profile 3 (by removing the password) and defines the password "Manager" for access profile 4.

Granting access

```
GRANT privilege ACCESS
  ON target_list
  TO acl_list;
```

Where privilege is one of:

READ	Allow read access
WRITE	Allow read and write access
ALL	Same as WRITE

The **`target_list`** is either **`ALL`** or a list of data set identifiers separated by a comma (,). If **`ALL`** is specified, the privilege will be granted to all data sets.

The **`acl_list`** is a list of access profile numbers separated by a comma (,). The keyword **`PUBLIC`** may be used as a synonym for access class 0.

For example:

```
GRANT READ ACCESS
ON Customers,Parts,Orders,Order-Entries
TO PUBLIC;
```

This will allow read access to the data sets Customers,Parts,Orders and Order-Entries to everyone.

```
GRANT WRITE ACCESS
ON Customers,Parts,Orders,Order-Entries
TO 1,2,3;
```

Write access to the given sets is enabled for members of the access class 1, 2 or 3.

Revoking access

```
REVOKE privilege ACCESS
  OF target_list
  FROM acl_list;
```

Where privilege is one of:

READ	Deny read access
WRITE	Deny read and write access
ALL	Same as WRITE

The **`target_list`** is either **`ALL`** or a list of data set identifiers separated by a comma (,). If **`ALL`** is specified, the privilege will be revoked from all data sets.

The **`acl_list`** is a list of access profile numbers separated by a comma (,). The keyword **`PUBLIC`** may be used as a synonym for access class 0.

For example:

```
REVOKE ALL ACCESS
OF Budget
FROM PUBLIC;
```

This revokes any access to the data set Budget for members of the access profile PUBLIC.

```
REVOKE WRITE ACCESS
OF Parts
TO 1,2;
```

This statement revokes write access on data set Parts for members of access pro files 1 and 2.

Creating a new data item

```
CREATE ITEM
   Ident, [count] type [ (format) ];

CREATE ITEM {
   Ident, [count] type [ (format) ];
   ...
}
```

`Ident` is the item name, `count` is the optional subitem count, `type` is the item type, `format` is an optional format number.

If the statement is enclosed in curly braces, multiple items can be defined.

Item type is the same as defined by the schema processor and must be one of:

I, I2	Allow read access
I4, D	Allow read and write access
R4, S	Short REAL
R8, L	REAL
Xn	String, size specified by n must be even

The format number may be an arbitrary number. It may be retrieved using DBINFO and is used by the QUERY program to define the item output format.

The QUERY format number is composed by adding the various values, which are given after the plus sign.

Table 2 **Query format numbers**

Bits	Field description	Field value	Description	add value
0	Item Protection	0	Item value may be changed in QUERY	0
		1	Item is write protected	1
1-2	Item type	0	default	0
		1	date type (days since 1972)	2
		2	Currency	4
		3	Undefined	6

Table 2 **Query format numbers**

Bits	Field description	Field value	Description	add value
0	Item Protection	0	Item value may be changed in QUERY	0
3	Item spacing	0	default	0
		1	Comma every 3 digits	8
4-6	Post decimals	0	default	0
		1	FIXED 0	16
		2	FIXED 1	32
		3	FIXED 2	48
		...		
		7	FIXED 6	112

For example:

```
CREATE ITEM
  Call-Id, D (2);
```

This defines the data item Call-Id of type 'D' with format number 2.

```
CREATE ITEM {
  Call-Id,   D;
  Call-Date, D (2);
  Call-Desc, 2X40;
}
```

This defines the data items **`Call-Id`**, **`Call-Date`** and **`Call-Desc`**.

Changing data item definition

```
ALTER ITEM Ident {
  [NAME   = Ident;]
  [TYPE   = [count] type [(fmt)]; ]
  [TYPE   = (fmt); ]
}
```

Ident is the item name, **count** is the optional subitem count, **type** is the item type, **format** is an optional format number.

The subitem count cannot be changed, if the data item is used as a search item or is included in an index item. The length of a string item cannot be reduced if it is used as a search item.

Changing the item type does will convert all values to the new item type.

Multiple item attributes must be enclosed in curly braces.

For example:

```
ALTER ITEM Phone-No
   TYPE  = X12;
```

This does change the type of the item **Phone-No** to an eight character string.

```
ALTER ITEM Phone-No {
  NAME = Phone;
  TYPE = 2 X20;
}
```

This does change the item name from **Phone-No** to **Phone**, the subitem count to two and the item type to a twenty character string.

Creating a new index item

```
CREATE IITEM
  Iident = Ident[:length ] ...;
```

```
CREATE IITEM {
  Iident = Ident[:length ] ...;
  ...
}
```

Iident is a new index item name, **Ident** is an existing item name and the optional **length** clause specifies a different index segment length. The index item name may not be defined as an item. Up to 7 index segments can be specified, separated by a comma (,).

If enclosed in curly braces, multiple index items can be defined.

For example:

```
CREATE IITEM
  ICall-Id = Call-Id;
```

This defines a new index item named ICall-Id using the value of the data item Call-Id.

```
CREATE IITEM
   ICall-Id = Call-Date, Call-Id;
```

This defines a new index item named **ICall-Id** combining the items **Call-Date** and **Call-Id**.

Changing data item definition

```
ALTER IITEM Ident {
   [NAME = Iident;]
   [TYPE = Ident[:length] ...;]
}
```

`Iident` is a new index item name, **`Ident`** is an existing item name and the optional **`length`** clause specifies an different index segment length. The index item name may not be defined as an item.

Up to 7 index segments can be specified, separated by a comma (,).

For example:

```
ALTER IITEM ICall-Id {
  NAME = IX-Call-Id;
  TYPE = Call-Id;
}
```

This changes the index item name from **`ICall-ID`** to **`IX-Call-Id`** and defines that the index is built using the value of the data item **`Call-Id`**.

Creating a new data set

```
CREATE SET SetIdent type {
   Ident [(MasterSetIdent)];
   ...
}
```

`SetIdent` is the name of a new data set, **`type`** is the data set type.

The optional **`MasterSetIdent`** is the name of a master data set.

Type must be one of:

M or MASTER	Master data set
A or AUTOMATIC	Automatic data set
D or DETAIL	Detail data set

For example:

```
CREATE SET Calls, D {
  Call-Id(Id);
  Call-Date;
  Call-Desc;
}
```

This creates the new detail data set **`Calls`** containing the data items **`Call-Id`**, **`Call-Date`** and **`Call-Desc`**.

The data item **`Call-Id`** is used as a search item with the master set **`Id`**.

Changing a data set:

The ALTER SET statement can be used to

- change data set properties
- add new data items to an existing data set
- add a new index to an existing data set

```
ALTER SET SetIdent
{
   [NAME = SetIdent;                  ]
   [CAPACITY = number;                ]

   [ADD ITEM {
     [Ident [(MasterSetIdent)];       ]
   }                                  ]

   [ADD INDEX
    Iident [/"collate"];              ]
}
```

SetIdent is the name of an existing data set, capacity is the capacity value.

Ident is an existing item name, the optional MasterSetIdent is the name of a master data set. A path can only be specified on new data sets.

Iident is an existing index item name, collate is the name of a collating sequence. The collating sequence name consists of a locale name and optional the modifier fold or nofold separated by an at (@) character. Please refer to the section "Collating sequences" for a description on collating sequences.

Changing Set properties

For example:

```
ALTER SET Calls {
  NAME = TheCalls;
  CAPACITY = 0;
}
```

This changes the data set name from Calls to TheCalls and the data set Capacity to zero.

Adding a data item

For example:

```
ALTER SET Calls {
  ADD ITEM {
     Call-Id(Id);
     Date;
     Code;
     Call-Desc;
  }
}
```

This adds four data items to the data set Calls. The Item Call-Id is used as a search item with an associated Master data set Id.

Adding an index

For example:

```
ALTER SET Calls {
  ADD INDEX ICall-Id;
  ADD INDEX ICode / "german@fold";
}
```

This will add two indices to data set Calls. The index ICode does use the collating sequence "german@fold".

Example script

```
### open database

DATABASE "db,DB" PASSWORD "Secret";

### passwords

ALTER PASSWORD {
 1 "READ";
 2 "WRITE";
}

### permissions

REVOKE ALL
OF ALL
FROM ALL;

GRANT READ
ON ALL
TO 1;

GRANT ALL
ON ALL
TO 2;

### items

CREATE ITEM {
 Call-Id,      D;
 Date,         D (2);
 Code,         X20;
 Description,  4X40;
}

### iitems

CREATE IITEM
 ICall-Id = Call-Id;

CREATE IITEM
 ICode = Code:10;

### set

CREATE SET Calls, D {
  Call-Id;
  Date;
  Code;
  Call-Desc;
}

ALTER SET Calls
{
   ADD INDEX ICall-Id;
   ADD INDEX ICode / "german@fold";
```

}

Interactive usage

The following general rules apply:

- If a Push button starts with a number, you can press the equivalent function key as an accelerator. For example **`[8. Exit]`** can be triggered with the function key F8.
- An underlined character indicates that the associated function can be reached by pressing function key F2 and the underlined character.
- An entry in a selection list can be selected by pressing the RETURN key or by pressing the space bar.
- There is currently no functionality associated with the help push button.

The Entry dialog

This dialog appears if dbutil is executed in interactive mode. The following dialog elements are present:

Database	The database name and path
Maintenance Password	The database maintenance password
Log	Shows the log window
Accept	Open the database
Exit	Exit the program

The Main dialog

The following dialog elements are present:

Log	Shows the log window
Exit	Exit the program. If there are unsaved changes, the exit dialog will appear.

The following functions can be selected from the selection list:

- Change maintenance password
- Change password or access
- Data Items
- Index Items
- Data sets
- Add Data Item to Set
- Add Index Item to Set
- Print Schema definition
- Analyze changes
- Apply changes

The Log dialog

The log windows holds a log of the most recent performed operation. If an error is encountered, the log dialog pops up automatically. The following dialog elements are present:

Close	Close the log window

The Exit dialog

The exit dialog pops up, if you are about to leave the program while while your changes have not been saved. The following dialog elements are present:

Save changes	If you select this checkbox and enter a file name in the following field, your changes are saved in a session script.
Accept	Exit the program. If selected, your changes are saved to a session script file.
Cancel	Don't exit the program, return to main dialog.

The Maintenance Password dialog

The following dialog elements are present:

Accept	Save changes.
Cancel	Discard changes, return to main dialog.

The Password dialog

This shows a list of all database access classes along with the password and an access summary. The first access class is PUBLIC which is used, if no password is provided to DBOPEN or no password is defined for the database. The R gives the number of data sets, this access class has READONLY access t, the W gives the number of data sets, this access class has READ/WRITE access to.

By selecting an access class, the access class properties dialog is opened. The following dialog elements are present:

Cancel	Discard changes, return to main dialog.

The Access class properties dialog

In the Access Class properties dialog, you can change the password or the access rights associated with an access class. The following dialog elements are present:

Password	Change the password associated with access class. If the password is empty, the access class becomes disabled. You cannot assign a password for the PUBLIC access class.
Read/Write	Select this to select or modify the data sets, the current access class has READ/WRITE access to.
Read only	Select this to select or modify the data sets, the current

	access class has READ ONLY access to.
Log	Shows the log window
Accept	Save changes, return to password dialog.
Cancel	Discard changes, return to password dialog.

To change the access rights for the current access class, select the Read/Write or Read Only push buttons. This opens the Data Set Selection dialog. It will display the list of data sets. In front of the data set name is a checkbox. If selected, the specified access is allowed for the current access class. If you select a data set, the access is changed.

The Data Item List dialog

This dialog shows all or a subset of the data items in your data base. By selecting a data item, the Data Item Properties dialog is opened. The following dialog elements are present:

Filter	The value entered in this field is used to select, which data items should be displayed. If this field is not empty, only data matching items are displayed
By Set	If selected, the value of the Filter field is considered a data set name. Only data items present in the selected data set are displayed.
Log	Shows the log window
Create	Opens the Item properties dialog to create a new data item.
Close	Close window, return to main dialog.

The Item Properties dialog

The following dialog elements are present:

Item Name	The data item name
Item Count	Number of elements
Item Type	Select an item type
Size	This is only used if the item is of type String. This is the maximum number of characters this item can hold. Size must be an even value.
Item Format	The item Format. This information can be used by QUERY or other programs to format their output. Please refer to the table "Query Format Numbers" for more information.
Log	Shows the log window
Accept	Save changes.
Cancel	Discard changes, return to previous dialog.

The Index Item List dialog

This dialog shows all or a subset of the index items in your data base. By selecting a data item, the Index Item Properties dialog is opened. The following dialog elements are present:

Filter	The value entered in this field is used to select, which index items should be displayed. If this field is not empty, only data matching items are displayed
By Set	If selected, the value of the Filter field is considered a data set name. Only data items present in the selected data set are displayed.
Log	Shows the log window
Create	Opens the Item properties dialog to create a new index item.
Close	Close window, return to main dialog.

The Index Item Properties dialog

The following dialog elements are present:

Name	The index item name
Segment / Item	Data item name for this index segment
Segment / Length	If this is a string item, only a leading part of the item may be used in the index.
Log	Shows the log window
Accept	Save changes.
Cancel	Discard changes, return to previous dialog.

The Data Set List dialog

This dialog shows all or a subset of the data sets in your data base. By selecting a data set, the Data Set Properties dialog is opened. The following dialog elements are present:

Filter	The value entered in this field is used to select, which data sets should be displayed. If this field is not empty, only matching data sets are displayed
Log	Shows the log window
Create	Opens the data set properties dialog to create a new data set.
Close	Close window, return to main dialog.

The Data Set Properties dialog

The following dialog elements are present:

Set Name	The data set name
Set Type	Select a set type

Capacity	The data set capacity.
Log	Shows the log window
Accept	Save changes.
Cancel	Discard changes, return to previous dialog.

The Data Set Entry dialog

This dialog is opened by selecting one of the following entries from the main dialog after selecting an Data Set from the Data Set Selection Dialog.

- Add Data Item to Set
- Add Index Item to Set

The dialog display the data or index items for the selected data set. The following dialog elements are present:

Log	Shows the log window
Add Item	Opens the Data Item or Index Item dialog.
Close	Close window, return to main dialog.

The Print Schema Definition dialog

The following dialog elements are present:

Write to file	Output Schema definition to file if selected.
File	File name and path.
Print to HP-UX printer	Output Schema definition to printer if selected
Printer	Printer name (and additional options).
Lines/Page	Number of lines / Page. Use zero to avoid no pagination.
Log	Shows the log window
Accept	Create Schema definition.
Cancel	Return to main dialog.

The Analyze dialog

The log windows holds a summary of all pending operations. The following dialog elements are present:

Close	Close the window

The Apply dialog

The exit dialog pops up, if you are about apply your changes to your database. The following dialog elements are present:

Save changes	If you select this checkbox and enter a file name in the following field, your changes are saved in a session script.
Analyze	This pops up the Analyze dialog. If summarizes the pend-

	ing changes.
Accept	Apply changes to your database. If selected, your changes are saved to a session script file.
Cancel	Don't apply changes, return to main dialog.

DBPATCH utility

If the database ROOT file contains different information than the data sets (e.g. number of records), the database is considered corrupted and a status word -94 is returned by DBOPEN. This may be the result of a power failure or a kill -9 of the daemon in the "right" moment.

The recommended procedure to recover from a corrupted database is to reload the database from a backup, because this will guarantee a fast and consistent recovery. However, there are situations The only supported way to repair your database is to export it using dbexport, erase it using dberase and re-fill it using dbimport utility programs.

The dbpatch utility can be used for two different purposes:

- If started without any options, it indicates which datasets are corrupted.

  Sample output of the dbpatch utility. The mark at the end of a line indicates a corrupted dataset:

```
                                     EXPECTED         REAL
DATA SET                   CAPACITY RECLEN ENTRIES  RECLEN ENTRIES
---------------- -- - --------- ------ --------- ------ ---------
INTERPRET        01 A       100     32        68     32        69 <
NAME              02 A       100      64        50      64        50
INHABER           03 M       200      76       161      76       161
DISC             04 D       500    128      2429    128      2430 <
```

- If started with the **-w** option it patches the ROOT file.

  THIS DOES NOT FIX ANY CORRUPTED DATA, it will simply suppress the status error.
  However, this may be necessary, if time does not permit you to export/import the database.

NOTE: The use of the dbpatch utility with the write option is UNSUPPORTED and may result in unpredictable behavior, due to errors in the "links" between data records or data sets (Master/Detail). You should repair a corrupted database using dbexport/dbimport as soon as possible.

DBIMPORT, DBEXPORT field separator

New commandline options:

-f sep - field separator, default is ','

The -f commandline option defines a different field separator.

The default field separator is a comma (',').

Dbexport/dbimport utility programs may be used to export data from a HP Eloquence database to an external application or to import data into a HP Eloquence database.

The -f commandline option makes it possible to define a different field separator character.

DBIMPORT, DBEXPORT performance enhancement

The dbimport and dbexport utility programs use the regular database runtime to access the database. A remarkable amount of time is spent on synchronization of processes for database access, while it's most often acceptable, that these programs use exclusive access.

With HP Eloquence A.05.01 there are special dbimport and dbexport utility programs called dbimports and dbexports, which use a optimized database runtime. Performance tests proved a performance gain of 30% to 50%.

Remote database access

Starting with HP Eloquence release A.04.02, network access to a remote database is supported. The remote database facility may be used with HP Eloquence or the HP Eloquence database library. The remote database service is available for the HP9000 and the PC platform.

NOTE: Please refer to *HP Eloquence database manual*, appendix E for a detailed description of the HP Eloquence database library functions. This document will focus on differences, specific to the PC platform only.

This section describes how to use the HP Eloquence remote database service the HP9000 and the PC platform.

Overview:

HP Eloquence does not allow an access via NFS, because data base access synchronization is done via shared memory, which can't be shared over a network. So any (write) access using NFS services would destroy the database.

HP Eloquence provides a network database daemon, which will provide access to a database via network. The network daemon will intercept requests from the network, process them and transmit results back to the invoking process.

The network daemon must be started on each machine, on which a database resides, which should be accessible via network. HP Eloquence must be installed on each machine, where a network daemon should be started.

A remote access version of the database library has been provided which will allow remote access to Eloquence database via network from 'C' language. HP Eloquence does not need to be installed on a remote system in order to use the remote library.

Prerequisites:

- HP Eloquence version A.04.02 or later
- HP-UX 9.0 or later
- HP-UX ARPA services

Installation:

The network daemon runs with a fixed user id. This is a potential security problem if it runs as 'root', because NO password must be specified to access ANY data in ANY database on the system.

The suggested solution is to either create a new login e.g. 'remote' or to use an existing login to start the network server instead of 'root'.

You must modify your system configuration in order to use or provide access to a database via network:

1. Modify the file /etc/services on the system where the database resides and on the system, from which you want to access the database. This will provide a mapping from a service name and a port number.

   Add a line like the following to your /etc/services

   ```
   eloq  8000/tcp  # HP ELOQUENCE eloqnd
   ```

   where eloq is the service name, 8000 is a port number and tcp is the protocol name. All characters after # are comment.The entries must not be preceded by a blank space.

   Check the port number against the following:

   - DON'T use a port number less than 1000
   - DON'T use a port number that is used by another service
   - DON'T use a port number between 5800 and 6100

   See the services(4) manual page for more information.

2. Start eloqnd.

   Start the network daemon from within your /etc/rc just after eloqd. Add a line like the following in the 'localrc' part of your /etc/rc file:

   ```
   su remote -c /opt/eloquence/bin/eloqnd
   ```

   where remote is the user login.

Remote access:

To access a remote database from within HP Eloquence, you have to define a VOLUME in your eloq.config or .eloqrc which will map a logical volume name to a remote path.

e.g. **`VOLUME RDB :RDB Remote:/opt/application`**

Any access to a database **`DBNAME,RDB`** will be performed using the network daemon on the system **`Remote`**.

You must provide an absolute path to the directory where the database resides, because there is no current directory.

There is a new status code for HP Eloquence database: A status of -700 indicates that a remote operation failed. Status word 10 describes the kind of failure:

- a negative number indicates an internal problem
- a positive number indicates a system error number. See errno(2) and /usr/include/sys/errno.h for details.

Compatibility:

Accessing a remote database should be fully transparent, except that all status words will be initialized with zero prior to remote database access (they remain unchanged accessing local database).

Remote access library:

We provide an additional library, which allows remote access to an HP Eloquence database from a system, where no HP Eloquence is installed.

To compile with a "remote only" database library, you have to define REMOTE_ONLY before including idb.h. You should link with libreloq.a instead of libeloq.a.

Code skeleton:

```
#define REMOTE_ONLY
#include <idb.h>
...
```

To link:

```
cc ... -l reloq
```

Remote-database-access from PC platform

This chapter describes how to use the HP Eloquence remote database service on the PC platform.

Please read the Installation and Configuration-chapter for information on installation and PC platform prerequisites.

NOTE: Please refer to *HP Eloquence database manual*, appendix E for a detailed description of the HP Eloquence database library functions. This document will focus on differences, specific to the PC platform only.

To use the HP Eloquence remote database access from the PC platform, the file **`ELOQDB.DLL`** must be installed either in your **`WINDOWS`** directory, or in a directory, which is referenced in the **`PATH`** environment variable.

You also need a **`WINSOCK.DLL`** installed. This DLL provides the network connection. It will be loaded by ELOQDB.DLL.

Specifying remote database

To access a database on a remote server, you have to specify the hostname and an absolute path name, where the database is located on the server, separated by a colon (:).

For example:

If your HP 9000 has the hostname **`server`** and the database, you want to access is **`/var/db/XXDB`**, you have to specify the following database name:

```
server:/var/db/XXDB
```

The **`eloqd`** and **`eloqnd`** daemons must be started on the server. The remote hostname must be contained in your **`HOSTS`** file and the service name eloq (defined in your **`SERVICES`** file) must use the same port numbers on your local PC and remote server.

Character set

All strings in the database are automatically converted from Roman8 character set to ISO 8859/1 character set used by MS Windows.

NOTE: Not all characters are convertible between Roman8 and ISO 8859/1 character sets. Some characters are unique to a single character set. If you packed some numeric data into a string variable, the content will probably be garbled.

Common problems

Network software not loaded

You either did not load the required network software or your network software is not compatible.

Connection refused

Remote host does not accept the connection. Either there is no server (eloqnd) for the requested service running on the remote host, or you installed different port numbers for the eloq service on your local PC than on the remote host.

Service eloq (tcp) not found in your SERVICES file

You did not install the eloq service name in your **`SERVICES`** file.

Host *name* not found in your HOSTS file

You did not define the host name *name* in your **`HOSTS`** file.

Development environment

The DLL providing HP Eloquence remote database access has been built using the Microsoft Visual C/C++ compiler revision 1.5. Your development environment must be compatible.

Coding considerations

You may easily migrate your HP-UX source code to a PC platform. It's compatible with the following exceptions:

- The status array must be of type **`long`** on the PC platform, whereas it may be either int or long on the HP-UX platform.
- There is no stdout or stderr on MS Windows. Fatal error messages are displayed in a dialog box.
- Only remote access is possible through the use of ELOQDB.DLL. Local operation is not supported.

To compile a program using the remote database access, you have to define the **`REMOTE_ONLY`** symbol to the preprocessor and link it with the **`ELOQDB.LIB`** import library.

Database locking

Selective DBUNLOCK

The HP Eloquence A.05.10 DBUNLOCK statement and the idb_unlock() function have been enhanced to support a selective unlock.

In addition to the DBUNLOCK mode 1 which does release all locks for a database, there are the new mode values 3 and 5, which release a specific lock as specified by the qualifier argument.

The following DBUNLOCK mode values are now supported:

Mode	UNLOCK Operation
1	Unlock database. All locks for the database are released.
3	Unlock dataset. A lock mode 3/4/13/14 is released. The qualifier argument must match the DBLOCK argument.
5	Unlock predicate. A lock mode 5/6/15/16 is released. The qualifier argument must match the DBLOCK argument.

In addition to the "official" modes above, DBUNLOCK also accepts and translates the following mode values:

Mode 2/11/12 is mapped to 1
Mode 4/13/14 is mapped to 3
Mode 6/15/16 is mapped to 5

This makes it possible to use the same DBLOCK and DBUNLOCK modes.

Conflicting Item lock resolution

The LOCK CONFLICTING ITEM configuration directive makes it possible to configure, how predicate locks with conflicting items are resolved.

Former HP Eloquence revisions rejected a predicate lock with a conflicting item, because this could lead to a situation, where two processes own a lock on an overlapping subset of data. If LOCK CONFLICTING ITEM is set to 1, predicate locks with conflicting items are granted. However any write attempt to data where another process owns a lock will result in a status error -12.

Please refer to the HP Eloquence configuration section further information.

Collating sequences

Due to a compatibility problem with the HP-UX 10.x NLS implementation, collating sequences used by the database are no longer read from the operating system. Instead, HP Eloquence provides its own collating sequences, extracted from HP-UX 9.x.

In order to be recognized by HP Eloquence, collating sequences must be present in the directory **`/opt/eloquence/lib/nls/collate`**.

5

HP Eloquence DLG enhancements

This chapter covers all changes to HP Eloquence dialog functionality since the initial documentation in the A.04.00 manual.

- New attributes
- DLG NEW statement
- DLG LOAD statement

New attributes

HP Eloquence A.05.10 supports the following new dialog attributes:

Common Attributes

The fgc, bgc Attributes

.fgc Define dialog-object “foreground” enhancements
.bgc Define dialog-object “background” enhancements

Both attributes accept a numerical argument, which specifies the enhancement by an additive value:

0	use default enhancement
+2	blinking video
+4	underline video
+8	half-bright video

A value of 16 will disable default enhancements.

For example:

```
DLG SET "Dialog.EditText.fgc",9
```

will cause the addressed EditText to be displayed reverse/half bright video instead of the default underline enhancement.

The kbind Attribute

The kbind attribute makes it possible to override the default keyboard handling for a specific object by assigning a rule for a key value.

For example:

The following statement assigns a rule value of 103 for a PageDown key (which has a key code of 338):

```
DLG SET "Object.kbind[338]",103
```

NOTE: The HP Eloquence **KBCODE** statement displays the code for a given key.
It is not recommended to make frequent use of the kbind attribute since this may lead to confusion due to nonstandard keyboard behavior.

The kbind attribute affects the actual object and its child objects. If you define a kbind attribute for a GroupBox, it is also active for all child objects. A different definition in a child object has priority over a parent definition.

The following GET and SET operations on the kbind attribute are defined:

- If a zero is assigned to a kbind attribute or a kbind attribute with index zero, all key assignments are removed. For example:

```
DLG SET "Object.kbind",0
DLG SET "Object.kbind[0]",0
```

- If a zero value is assigned to a kbind index, the specified key assignment will be removed. For example:

```
DLG SET "Object.kbind[338]",0
```

- If a nonzero value is assigned to a kbind index, the specified key will be assigned with the given rule value For example:

```
DLG SET "Object.kbind[338]",103
```

- The value returned by a kbind attribute or a kbind attribute with index zero is the number of kbind assignments. For example:

```
DLG GET "Object.kbind",N
DLG GET "Object.kbind[0]",N
```

- The value returned by a kbind attribute with an index N is the Nth key assignment. For example:

```
DLG GET "Object.kbind[1]",N
```

The following example program will demonstrate the various DLG GET/SET operations on the kbind attribute:

```
DLG SET "Object.kbind",0          ! Clear all key assignments
DLG SET "Object.kbind[65]",20     ! 'A' returns 20
DLG SET "Object.kbind[32]",21     ! space returns 21

DLG GET "Object.kbind",Nkeys      ! Get number of values
DISP "Nkeys=";Nkeys

FOR I=1 TO Nkeys                  ! Get all values
   DLG GET "Object.kbind["&VAL$(I)&"]",K
   DISP "kbind["&VAL$(I)&"]=";K
NEXT I
```

The following keys can not be reassigned, because they are processed internally:

- The ESC key (27)
- The BREAK key (0) and the ^Y key (25)
- The ^L key (12)
- The ^N key (14)
- The Insert key (331)
- The key assigned to the alt attribute.

NOTE: This attribute is not available with Dialog Manager.

GroupBox Attributes

HP Eloquence A.05.10 supports the **`sensitive`** attribute in a GroupBox. If a GroupBox is not sensitive, all of its child-objects are insensitive, too (they can not get the keyboard focus). A GroupBox is sensitive by default.

ListBox Attributes

The topitem Attribute

The ListBox object type supports the topitem attribute. The topitem attribute indicates the first visible line in a ListBox.

When set, the ListBox is scrolled appropriately.

NOTE: This attribute is not available with Dialog Manager.

The line Attribute With Index

The ListBox object type supports an additional index specified for the line attribute.

Without an index, the line attribute will get or set the line as indicated by the cy attribute. If an index is specified, the specified line is got or set. The first line is specified by 1.

For example:

```
DLG SET "List.line[12]","Text"
```

The insert Attribute

.ins

The ins attribute is of type string. It inserts the given string before the current line.

NOTE: ListBox always inserts complete lines.

The Delete Line Attribute

.delln

The delln attribute is of type integer. It deletes the given number of lines starting at the cursor line.

EditText attributes

The topitem Attribute

The EditText object type supports the topitem attribute.

The topitem attribute indicates the first visible line in an EditText. When set, the EditText is scrolled appropriately.

NOTE: This attribute is not available with Dialog Manager

The line Attribute With Index

The EditText object type supports an additional index specified for the line attribute.

Without an index, the line attribute will get or set the line as indicated by the cy attribute. If an index is specified, the specified line is get or set. The first line is specified by 1.

For example:

```
DLG SET "Edit.line[12]","Text"
```

Insert and Overwrite

EditText objects now support both modes: insert and overwrite

Previously, if text was entered into an EditText, it was always inserted. This was often unexpected to HP Eloquence users. Now EditText is set to overwrite mode by default. You may toggle overwrite/insert mode using the INSERT key. The cursor form will change to indicate insert/overwrite mode.

The writefile Attribute

The EditText objects support the following additional attributes:

.writefile

The writefile attribute is of type string. It specifies a file name, where the contents of the EditText are written to.

Using this attribute, you can write your own Text Editor very simply:

```
DLG SET "dialog.edit.file","/tmp/text"          ! read file
DLG DO "dialog",R                               ! handle dialog
DLG SET "dialog.edit.writefile","/tmp/text.new"  ! write file
```

The insert Attribute

.ins

The ins attribute is of type string. It inserts the given string at the current cursor position.

The Delete Line Attribute

.delln

The delln attribute is of type integer. It deletes the given number of lines starting at the cursor line.

Dialog Object Attributes

The dialog object type supports the alt attribute. If it is nonzero, it is the key number of the key to be used as an ALT key in this dialog. The alt attribute must be defined in order to enable keyboard accelerators in the dialog (see below).

Keyboard Accelerators

If a '&' character is contained in the label text of a StaticText, CheckBox, RadioButton or PushButton object type, the following character will be displayed underlined and used as a keyboard accelerator.

NOTE: If you want to display a “&” in a StaticText, you must not define the alt attribute.

If the alt attribute is enabled in the dialog, pressing the ALT key (as specified by the alt attribute) and a keyboard accelerator causes the focus to change to the first object defining the accelerator key.

If the object type is StaticText, keyboard focus is changed to the next available object. If the object type is of type CheckBox, RadionButton or PushButton, the object is selected.

If pressing a key which is not accepted by the current object (the object which has the focus), it will be used as an accelerator.

For example, pressing an alphanumeric key while the current object is of type PushButton will cause an accelerator lookup.

For example:

```
Dialog Sample {
   ...
   .alt = 266    # Alt key is F2

   StaticText Label1 {
      ...
      .text = "Label &1"
```

```
        }
        EditText Edit {
            ...
        }
        PushButton OK {
            ...
            .text = "&OK"
        }
    }
```

Pressing the keys F2 and 1 will change the focus to the EditText “Edit”. Pressing the keys F2 and O will trigger the PushButton “OK”.

DLG NEW enhancements

The DLG NEW statement does dynamically creation of an dialog object.

You have to specify the object path and an object type.

It’s also possible to specify an object path instead of the object type. The specified object and all it's children are copied to the given target path.

With this functionality you are able to use *Models* in your dialogs. The dialog description file contains the *Model* and *Dialog* description, as two dialogs.

For example:

```
Dialog Model {
   ...

   PushButton OK {
      .text = "&OK"
      .rule = 1
   }
   PushButton HELP {
      .text = "&HELP"
      .rule = -1
   }
}

Dialog Sample {
   ...

   Model.OK Ok {
      .x = 10
      .y = 10
   }
   Model.HELP Help {
      .x = 20
      .y = 10
   }
}
```

The PushButtons “OK” and “Help” will be created in the dialog Sample.

All attributes are derived as specified in the dialog Model.

It’s also possible to dynamically create objects from within HP Eloquence:

```
DLG NEW "Sample.OK","Model.OK"
```

This creates the object OK in the dialog sample as specified by the object Model.OK.

NOTE: For further information concerning this statement read the A.04 HP Eloquence Manual.

DLG LOAD enhancements

In addition to dialog resource specifications, there are two additional directives supported in a dialog resource file:

DLG LOAD debugging

The DLG LOAD statement checks the syntax of the dialog file and reports a self explaining error message to the screen, if it is executed interactively.

For example:

```
DLG LOAD "dialog.dlg"
LINE #5: DLG NEW "Test.select","Lisrbox" failed
```

The include directive

The include directive makes it possible to have common definitions among dialog files (including context specific definitions).

```
include "file"
```

This will include the given file. If a relative path is specified, the include file is loaded relative to the path, the current dialog file is loaded from.

The file name may also contain environment variables. Environment variables are expanded in order to locate the file.

For example:

```
include "$HOME/sample.inc"
```

will include the file sample.inc from the home directory.

```
include "sample.inc"
```

will include the file sample.inc from the same directory as the dialog file.

If the first character is a question mark ('**?**'), DLG LOAD will not fail if the include file is not present.

For example:

```
include "?$HOME/sample.inc"
```

will try to include the file sample.inc from the home directory.

Include file nesting is limited to two levels.

The define directive

The define directive makes it possible to use Variables in the dialog file.

```
Name = value
```

This will associate Name with value. This name may subsequently be used to ref erence the associated value.

If *Name* has already been defined, it will be updated with the current value.

For example:

```
# border types
Bd_None    = 0
Bd_Thin    = 1
Bd_Thick   = 2
Bd_shadow  = 3
Bd_rshadow = 4

ALT_KEY    = 266
TITLE      = "Sample string"

Dialog sample {
   ...
   .border = Bd_Thick
   .alt = ALT_KEY
   .title = TITLE
   ...
}
```

NOTE: For further information concerning this statement please refer to the A.04.00 HP Eloquence manual.

FRM2DLG conversion utility

FRM2DLG converts HP Eloquence FORM files into HP Eloquence dialog files. It will create a dialog-file on the MSI volume with the same name as the FORM file. After the conversion process has finished, the dialog-file will be displayed on the screen. Press the function key F8 to terminate.

It's located in the **`/opt/eloquence/share/contrib/`** directory.

As its location in the contrib directory indicates, FRM2DLG has been provided as a template, that you may customize to fulfill your specific needs.

NOTE: This program is not supported and provided only as an example.

6

HP Eloquence Graphical User Interface

The graphical user interface is implemented through (platform dependent) dialog drivers. If you activate a dialog driver, all HP Eloquence DLG statements are no longer executed by HP Eloquence, but passed to the specified driver.

The driver will map the HP Eloquence DLG statements to ISA Dialog Manager intrinsic calls which will handle your display.

Overview

You may simply activate a driver using one of the following DLG SET statements:

```
DLG SET ".driver","motif"
DLG SET ".driver","alpha"
```

The first example above will redirect all DLG statements to the motif driver, while the second example will redirect all DLG statements to the alpha driver.

```
DLG SET ".driver","@client"
```

The example above will redirect all DLG statements to the network driver running on the system named *client*.

NOTE: Using the network driver requires the **RUNSRV** utility to be running on the *client* system.

NOTE: Although it's possible to use HP Eloquence dialog files (they are converted internally at runtime) it's strongly recommended to convert them to Dialog Manager format due to performance considerations.

Once converted, you're able to change the layout with the Dialog Manager graphical editor.

The Dialog is converted using the **CVDLG** utility:

```
cvdlg -driver motif sample.idm sample.dlg
```

This will convert the HP Eloquence dialog file *sample*.dlg to Dialog Manager dialog file *sample*.idm using the motif driver.

It's also possible to compile a Dialog Manager dialog file. This has some additional performance advantages during dialog file load time. Compilation is done through the Dialog Manager idm utility.

For Example:

```
idm +writebin sample.idc sample.idm
```

This will compile the Dialog Manager dialog file *sample*.idm into a file *sample*.idc.

Compiled Dialog Manager files are platform dependent.

NOTE: The idm utility program is *not* included with HP Eloquence. You must purchase the Dialog Manager product to be able to compile dialog files.

To be able to automatically select the appropriate dialog file, the DLG LOAD statement has been enhanced if a dialog driver has been activated. If you specify a file extension of “.dlg”, DLG LOAD will look for the following files (in this order):

.idc	compiled Dialog Manager file
.idm	Dialog Manager file
.dlg	HP Eloquence dialog file

The file extensions may be customized in the eloq.ini configuration file.

For Example:

```
DLG LOAD "sample.dlg"
```

will try to load “sample.idc” first, then “sample.idm” and at last “sample.dlg”.

NOTE: If we talk in the following documentation about dialog server we talk about software that runs on the system on which the user interface should be displayed. The dialog client is the HP Eloquence application, because it sends requests to the dialog server to handle the user interface.

NOTE: If we talk about systems, the server system is the system on which the HP Eloquence application runs, while the client is the system on which the user interface is displayed.

HP Eloquence Dialog Drivers

The graphical user interface is implemented through (platform dependent) dialog drivers. If you activate a dialog driver, all HP Eloquence DLG statements are no longer executed by HP Eloquence, but passed to the specified driver.

The driver will map the HP Eloquence DLG statements to ISA Dialog Manager intrinsic calls which will handle your display.

This document describes the extensions to HP Eloquence, if a dialog driver has been activated.

HP Eloquence supports the following attributes:

.driver	SET/GET dialog driver
.async	SET/GET driver communication mode

Starting a driver

To start a driver from within HP Eloquence, you issue the following command:

```
DLG SET ".driver","driver_spec [ini_section [arguments]]"
```

driver_spec:	*motif*	start the motif driver
	alpha	start the alpha driver
	@client	start the network driver on the system named *client*

ini_section: Optional name of a *user-defined section* in the eloq.ini file where the defaults can be overridden.

arguments: Additional arguments can optionally be specified here and will be passed-through to the Dialog Manager.

On driver start-up, the following tasks are performed:

1. The driver sets up the *Dialog Manager argument list* from the **`arguments`** specified in the **`DLG SET ".driver"`** statement, if any. The driver then reads the configuration items from the eloq.ini file.
2. If **`ini_section`** is specified in the **`DLG SET ".driver"`** statement, the driver reads additional configuration items from this *user-defined section* in the eloq.ini file.
3. If **`ini_section`** is specified in the **`DLG SET ".driver"`** statement, the driver searches this *user-defined section* in the eloq.ini file for an item named *Arguments*. If this item exists, its value is appended to the *Dialog Manager argument list*.

   Example:

```
[debug]
Arguments = -IDMtracefile /tmp/idmtrace
```

This enables an additional Dialog Manager argument which creates a trace file for debugging purposes. In order to activate this item, the name of this section must be specified in the **DLG SET ".driver"** statement, e.g.:

```
DLG SET ".driver","motif debug"
```

4 Finally, the composed *Dialog Manager argument list* is passed to the Dialog Manager runtime system start-up function.

NOTE: For a list of valid command line arguments, please refer to the *ISA Dialog Manager* documentation.

NOTE: For details about *configuration items* and *user-defined sections* please refer to chapter *Configuring HP Eloquence*, sections *Customize the ELOQ.INI File on the HP-UX System* and *Customize the ELOQ.INI File on the PC Platform*.

Setting (or re-setting) driver will result in an implied **DLG STOP**, this statement will reset the dialog driver.

For example:

```
DLG SET ".driver","motif"
DLG SET ".driver","alpha"
```

The first example above will redirect all DLG statements to the motif driver, while the second example will redirect all DLG statements to the alpha driver.

```
DLG SET ".driver","@client"
```

The example above will redirect all DLG statements to the network driver running on the system named *client*.

NOTE: For a more specific description of the drivers, please refer to the sections *Motif Driver*, *Alpha Driver* and *Network Driver* in this chapter.

NOTE: Using the network driver requires the **RUNSRV** utility to be running on the *client* system.

Driver ASYNC mode

The dialog client and server processes may communicate in two ways:

Synchronous Each request by the client results in a server response. Processes have to wait on each other.

Asynchronous Only specific requests result in a server response. Processes run concurrently.

Given the fact that server operations are normally successfully completed, a lot of communication overhead is generated. Using ASYNC mode, this overhead is reduced dramatically, resulting in much better response time. However, because the server does not necessarily notify the client immediately when a problem is recognized, the error message may be retrieved at any statement executed later.

To control the driver ASYNC communication mode, you must use the following command:

```
DLG SET ".async",mode
```

where mode is either 1 (ASYNC mode active) or 0 (SYNC mode active).

Only the network driver (client-server) can run in ASYNC communication mode, but all drivers accept the syntax above.

Only the following commands will use the ASYNC communication mode:

- DLG SET
- DLG STOP
- DLG NEW
- DLG DEL

During program debug time, it is recommended to use the SYNC mode. (DLG SET ".async",0). This is also required if you expect a command to fail. Default communication mode is SYNC mode.

It might have a performance advantage to use the ASYNC mode during a transfer of a bigger amount of data. e.g. filling a ListBox or a multiline EditText.

Recovering from runtime error in ASYNC mode

If you encounter a runtime error while communicating in ASYNC mode, it’s necessary to re-sync client and server processes again. To accomplish this you should issue a DLG SET ".async" with any mode.

Driver Reference

This chapter describes the drivers provided with HP Eloquence:

- Motif
- Alpha
- Network

Motif Driver

motif.drv is the HP Eloquence dialog driver for the Motif GUI. It must be installed in the directory **/opt/eloquence/lbin**.

Starting the Motif Driver

To start the motif driver from within HP Eloquence, you issue the following command:

```
DLG SET ".driver","motif"
```

This will redirect output to your display as defined in the **DISPLAY** environment variable. If the **DISPLAY** environment variable is not set or if you want to output to a different display you can specify the display address in the driver command using the **-display** argument.

The driver provides a method to lookup additional arguments in a *user-defined section* of your eloq.ini file. Using this method, you can maintain system-specific driver arguments at a centralized location (eloq.ini), so you can execute your HP Eloquence program in different environments without code changes.

Example:

```
[myoptions]
Arguments = -display client:0
```

This enables an additional Dialog Manager argument which specifies a different output display *client:0* where **motif.drv** will send its output to. In order to activate this item, the name of your *user-defined section* must be specified in the **DLG SET ".driver"** statement, e.g.:

```
DLG SET ".driver","motif myoptions"
```

NOTE: For details about *configuration items* and *user-defined sections* please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the HP-UX System.*

Driver arguments can also immediately be specified in the **`DLG SET ".driver"`** statement. This is useful for arguments which are closely related to **`motif.drv`** and not system-specific, e.g.:

```
DLG SET ".driver","motif myoptions -IDMfont 0 -IDMcolor 0"
```

These two Dialog Manager arguments set the font and color resources to *variant 0*, this is normally appropriate for **`motif.drv`**.

NOTE: If driver arguments are immediately specified in the **`DLG SET ".driver"`** statement, an *user-defined section* must be specified, too. However, if you specify a section name not present in the eloq.ini file, the *user-defined section* will be ignored.

NOTE: For a list of valid command line arguments, please refer to the *ISA Dialog Manager* documentation.

Alpha Driver

`alpha.drv` is the HP Eloquence dialog driver for text terminals such as HP70092 or VT220. It must be installed in the directory **`/opt/eloquence/lbin`**.

NOTE: The **`alpha.drv`** is different from HP Eloquence DLG since **`alpha.drv`** uses the Dialog Manager runtime functionality while DLG is a built-in HP Eloquence component. Normally, text terminals are directly attached to the system running HP Eloquence, so DLG performs better compared to **`alpha.drv`** due to the **`alpha.drv`** client-server protocol overhead. For this reason, it is recommended to use HP Eloquence DLG for text terminals unless you need native Dialog Manager functionality.

Starting the Alpha Driver

To start the alpha driver from within HP Eloquence, you issue the following command:

```
DLG SET ".driver","alpha"
```

This will redirect output to your display as defined in the **`DISPLAY`** environment variable. If the **`DISPLAY`** environment variable is not set or if you want to output to a different display you can specify the display address in the driver command using the **`-display`** argument.

The driver provides a method to lookup additional arguments in an *user-defined section* of your eloq.ini file. Using this method, you can maintain system-specific driver arguments at a centralized location (eloq.ini), so you can execute your HP Eloquence program in different environments without code changes.

Example:

```
[myoptions]
Arguments = -display /dev/tty0p6
```

This enables an additional Dialog Manager argument which specifies a different output display */dev/tty0p6* where **`alpha.drv`** will send its output to. In order to activate this item, the name of your *user-defined section* must be specified in the **`DLG SET ".driver"`** statement, e.g.:

```
DLG SET ".driver","alpha myoptions"
```

NOTE: For details about *configuration items* and *user-defined sections* please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the HP-UX System*.

Driver arguments can also immediately be specified in the **`DLG SET ".driver"`** statement. This is useful for arguments which are closely related to **`alpha.drv`** and not system-specific, e.g.:

```
DLG SET ".driver","alpha myoptions -IDMfont 2 -IDMcolor 2"
```

These two Dialog Manager arguments set the font and color resources to *variant 2*, this is normally appropriate for **`alpha.drv`**.

NOTE: If driver arguments are immediately specified in the **`DLG SET ".driver"`** statement, an *user-defined section* must be specified, too. However, if you specify a section name not present in the eloq.ini file, the *user-defined section* will be ignored.

NOTE: For a list of valid command line arguments, please refer to the *ISA Dialog Manager* documentation.

Network Driver

The HP Eloquence client-server network driver is implemented for the Microsoft Windows GUI. The name of this driver is **`DLGSRV`**.

Starting the Network driver

To start the network driver from within HP Eloquence, you issue the following command:

```
DLG SET ".driver","@client"
```

where *client* is the name of the system running Microsoft Windows where the user interface (dialog server) should run.

NOTE: *Client* must be defined in your **/etc/hosts** file. You may also use the IP address instead of the host name.

When starting the network driver, you can pass additional arguments to the Dialog Manager runtime system. The driver provides a method to lookup additional arguments in an *user-defined section* of your eloq.ini file. Using this method, you can maintain system-specific driver arguments at a centralized location (eloq.ini), so you can execute your HP Eloquence program in different environments without code changes.

Example:

```
[debug]
Arguments = -IDMtracefile C:\TMP\IDMTRACE.TXT
```

This enables an additional Dialog Manager argument which creates a trace file for debugging purposes. In order to activate this item, the name of your *user-defined section* must be specified in the **DLG SET ".driver"** statement, e.g.:

```
DLG SET ".driver","@client debug"
```

NOTE: For details about *configuration items* and *user-defined sections* please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform.*

Driver arguments can also immediately be specified in the **DLG SET ".driver"** statement. This is useful for arguments which are closely related to the network driver and not system-specific, e.g.:

```
DLG SET ".driver","@client myoptions -IDMfont 1 -IDMcolor 1"
```

These two Dialog Manager arguments set the font and color resources to *variant 1*, this is normally appropriate for the Microsoft Windows network driver.

NOTE: If driver arguments are immediately specified in the **DLG SET ".driver"** statement, an *user-defined section* must be specified, too. However, if you specify a section name not present in the eloq.ini file, the *user-defined section* will be ignored.

NOTE: For a list of valid command line arguments, please refer to the *ISA Dialog Manager* documentation.

Starting the Network Driver on the Remote System

The network driver must be started on the remote system in order to get connected to the server system running HP Eloquence. This is done either automatically or manually.

- **Automatic Start-up**

  If the *PortRange* configuration item defined in section *[dmclnt]* of the HP-UX eloq.ini file is nonzero, HP Eloquence expects the **RUNSRV** utility to run on the remote system. **RUNSRV** will then be used to start-up the network driver on the remote system.

  This is the recommended method to start-up the network driver.

- **Manual Start-up**

  If the *PortRange* configuration item defined in section *[dmclnt]* of the HP-UX eloq.ini file is zero, HP Eloquence will wait until the network driver is started manually on the remote system.

  If you choose this method to start-up the network driver, **DLGSRV** expects a command line argument of *-connect servername:portnumber* which enables **DLGSRV** to connect to the server system (running HP Eloquence).

  Using this nonstandard method to start-up the network driver is not recommended.

NOTE: Additional information is provided in section *The DLGSRV Utility* later in this chapter.

The RUNSRV Utility

RUNSRV is a utility program which is used by HP Eloquence to start the HP Eloquence network dialog server (**DLGSRV**) on a remote PC running Microsoft Windows. To achieve this, the **RUNSRV** program must be active on the remote PC.

Starting the RUNSRV Utility

If you start **RUNSRV.EXE** on the remote PC, it will minimize itself waiting for a request.

If you restore the **RUNSRV** window, a log of the recent operations performed by **RUNSRV** is displayed. If **RUNSRV** is unable to handle a request, a message box pops up.

NOTE: **RUNSRV** uses the **WINSOCK.DLL** on the PC to interface the network software.

Common Problems

`Network software not loaded`

You either did not load the required network software or your network software is not compatible.

`Service runsrv (tcp) not found in your SERVICES file`

You did not install the runsrv service name in your **SERVICES** file.

`Address already in use`

The port number you used by **RUNSRV** is already in use. This may happen if the port number is not unique or if you try to run **RUNSRV** a second time.

`WinExec failed (nn): Error Message`

RUNSRV was unable to perform the requested operation. Error number and message reflect a problem detected by Microsoft Windows.

The Remote PC ELOQ.INI File

RUNSRV uses the configuration information of the *[runsrv]* section in the eloq.ini file on the PC. This file is located in your **WINDOWS** directory.

NOTE: Please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [runsrv]* for details.

The HP-UX RunSrv.DLL

`RUNSRV` is not limited to start the network dialog server only. It may also be used to execute any command on a remote PC. The **`RunSrv.DLL`** provides the interface for HP Eloquence to communicate with **`RUNSRV`** on a remote PC.

The **`RunSrv.DLL`** is located in the **`/opt/eloquence/dll`** directory. It uses the configuration information of the *[runsrv]* section in the eloq.ini file on the HP-UX system. This file is located in the HP Eloquence configuration directory:

`HP-UX 9.x:` **`/opt/eloquence/etc/eloq.ini`**

`HP-UX 10.x:` **`/etc/opt/eloquence/eloq.ini`**

Using the **`RunSrv.DLL`**, you may use the **`RUNSRV`** functionality in your own programs:

- Load the **`RunSrv.DLL`**:

```
LOAD DLL Run,"/opt/eloquence/dll/RunSrv.DLL",1024
```

- Call the function **`RunSrv`** in the **`RunSrv.DLL`**. You specify the host name, the remote command string (to be executed on the remote host), an *INTEGER* variable to receive an error number and a string variable to receive an error message:

```
CALL DLL Run("RunSrv",Host$,Cmd$,Errn,Msg$)
```

- Shutdown the **`RunSrv.DLL`** if you don't need it any more:

```
DEL DLL Run
```

The following HP Eloquence error numbers are returned by the **`RunSrv.DLL`**:

- 2000 - Memory allocation failed
- 2001 - Invalid argument
- 2002 - Other error (see error string for details)

For an example see the HP Eloquence **`RUNSRV.PROG`** example program.

NOTE: Please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the HP-UX System, Section [runsrv]* for details.

The RUNSRV Network Protocol

`RUNSRV` uses the following TCP/IP network protocol:

Each message is preceded by a 4 byte length field in network order. The following structure is used to transmit a request:

```
struct Job {
   char type[4];/* type of request, \0 terminated */
   char data[1];/* additional data for request, variable size */
};
```

Currently, there are three job types supported:

DLG Start the HP Eloquence network dialog server (**DLGSRV**). Connection information transmitted by HP Eloquence is merged with the command line given in the eloq.ini file on the remote PC (configuration item *DlgSrvX* or *DlgSrv* in section *[runsrv]*).

RUN Executes a program on the remote PC. *Job.data* holds the command line to be executed, terminated by a null byte.

DDE Executes a Microsoft Windows dynamic data exchange (DDE) command on the remote PC. *Job.data* holds the DDE command string to be executed, terminated by a null byte.

NOTE: Normally, the **RunSrv.DLL** is used to communicate with **RUNSRV** using this protocol. Please refer to the former section *The HP-UX RunSrv.DLL* for details.

NOTE: Please refer to the next section *RUNSRV DDE Communication* for details about Microsoft Windows dynamic data exchange.

RUNSRV DDE Communication

RUNSRV has built-in support for Microsoft Windows dynamic data exchange (DDE). This can be used to remotely control Microsoft Windows applications from a HP Eloquence program.

If you intend to communicate with a Microsoft Windows application using DDE, this application must be currently running on the remote PC. If it is not, **RUNSRV** provides a method to automatically start it.

For each Microsoft Windows application you intend to communicate with using DDE, add a configuration item to the section *[modules]* of the eloq.ini file on the remote PC.

Example:

```
[modules]
WinWord = C:\WINWORD\WINWORD.EXE
```

This associates the *module identifier* of *Microsoft Word for Windows* with the program execution path. This enables **RUNSRV** to automatically start-up *Microsoft Word for Windows* if it's not currently running.

NOTE: If you want to communicate with any Microsoft Windows program using DDE, knowledge about the Microsoft Windows *module identifier* and the communication topics specific to the program is required. Normally, these informations are contained in the program's documentation. The Microsoft Windows *Task List* displays the *module identifiers* of all programs currently running.

NOTE: Please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [modules]* for details.

DDE communication is performed by calling the function **RunSrv** in the **RunSrv.DLL**. The call format is:

```
CALL DLL Run("RunSrv",Host$,Cmd$,Errn,Msg$)
```

This call format is described in section *The HP-UX RunSrv.DLL* prior in this chapter. The DDE command is specified in **Cmd$** in the following format:

```
Cmd$ = "DDE ModuleName Topic TransactionType Item Value"
```

DDE: This is the literal string "DDE" used to identify **Cmd$** as a DDE communication command string.

ModuleName: This is the Microsoft Windows *module identifier* of the application you want to communicate with.

Topic: This is a communication entity specific to the application you want to communicate with.

TransactionType: This defines the type of the DDE communication. It depends on application requirements for the given **Topic**.

The following **TransactionTypes** are implemented:

- **REQUEST**
- **POKE**
- **EXECUTE**

REQUEST is the only **TransactionType** that returns data. The only data type recognized is *CF_TEXT*.

Item: This is the communication item name. It depends on application requirements for the given **Topic**.

Value: This is only required for the **POKE Transaction Type** and specifies the data to be transmitted to the Microsoft Windows

application.
Do not specify **Value** for the other **TransactionTypes**.

These elements in **Cmd$** are separated by white space and must not have any white space separators inside.

On success, the **RunSrv** function sets **Errn** to 0. After a **REQUEST** transaction, **Msg$** contains the DDE response data provided by the Microsoft Windows application. All other **TransactionTypes** return **Msg$=""**.

On failure, **Errn** is set to -1 for DDE protocol errors (e.g. if the desired communication could not be established or the application did not recognize the given **Topic**) or 1 if a DDE protocol time-out occurred (e.g. a **REQUEST** transaction failed to receive the return data because the application program did not respond).

A literal description of the error code is returned in **Msg$**.

The following example will start-up Netscape and will cause it to load a specific document:

```
INTEGER Errn
DIM Msg$[256],Cmd$[100]

Cmd$="DDE Netscape WWW_OpenURL Request "
Cmd$=Cmd$&CHR$(34)
Cmd$=Cmd$&"http://www.msede.com/public/eloquence.html"
Cmd$=Cmd$&CHR$(34)
Cmd$=Cmd$&",,0xFFFFFFFF,0x0"

CALL DLL Run("RunSrv",Host$,Cmd$,Errn,Msg$)
```

NOTE: The **Item** format in the **Cmd$** above is specific to the Netscape application.
The last parameter **0x0** specifies if the document and/or image caches are to be used when loading the document. Netscape will not reload the document if it is already located in the document cache. If you want a document to be reloaded, change this parameter to **0x3** (where **0x1** means “ignore the document cache”, **0x2** means “ignore the image cache” and **0x3** combines these two options).

The DLGSRV Utility

`DLGSRV` is the HP Eloquence dialog server for the Microsoft Windows GUI. It runs on the remote PC and contains the Dialog Manager runtime functionality for Microsoft Windows.

Please refer to the chapters *Installing HP Eloquence* and *Configuring HP Eloquence* for details about installation and PC platform prerequisites.

Starting the DLGSRV Utility

Normally, **`DLGSRV`** is started automatically by the **`RUNSRV`** utility, which will provide the appropriate command line arguments. This is done when a **`DLG SET ".driver","@client"`** statement is issued. **`RUNSRV`** then uses the configuration item *DlgSrvX* or *DlgSrv* of the *[runsrv]* section in the eloq.ini file on the PC to compose the appropriate command line to start **`DLGSRV`**.

NOTE: Please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [runsrv]* for details about *DlgSrvX* and *DlgSrv*. Additional information is provided in *How to Run Multiple DLGSRV Simultaneously* later in the current section.

NOTE: **`DLGSRV`** uses the **`WINSOCK.DLL`** on the PC to interface the network software.

However, if the nonstandard method of manually starting **`DLGSRV`** is used, **`DLGSRV`** expects a command line argument of *-connect servername:portnumber* which enables **`DLGSRV`** to connect to the server system (running HP Eloquence):

```
DLGSRV.EXE -connect servername:portnumber [arguments]
```

-connect: This introduces the connection argument *servername:portnumber*.

servername: This is the host name of the server system (running HP Eloquence) **`DLGSRV`** shall connect with.

NOTE: You should provide an appropriate entry for *servername* in your **`HOSTS`** file on the remote PC. The location of your **`HOSTS`** file depends on your networking software.

portnumber: This is the port number **`DLGSRV`** will use to connect to the server system (running HP Eloquence).

arguments: Additional arguments can optionally be specified here and will be passed-through to the Dialog Manager.

NOTE: For a list of valid command line arguments, please refer to the *ISA Dialog Manager* documentation.

When **DLGSRV** starts up, it will minimize itself. If you restore the **DLGSRV** window, a message box pops up, which allows you to terminate the **DLGSRV** program. This may also be achieved by selecting *Close* from the system menu.

NOTE: You should not terminate **DLGSRV** this way since this may lead to loss of data and abnormal termination of the server program. **DLGSRV** will be terminated automatically by HP Eloquence when processing has finished.
However, to recover from program failure or communication breakdown this method of manually terminating **DLGSRV** provides a very useful “emergency exit”.

The Remote PC ELOQ.INI File

DLGSRV uses the configuration information of the *[dlgsrv]* section in the eloq.ini file on the PC. This file is located in your **WINDOWS** directory.

NOTE: Please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [dlgsrv]* for details.

How to Run Multiple DLGSRV Simultaneously

It is not possible to have more than one running instance of the HP Eloquence network dialog server for Microsoft Windows (**DLGSRV**). This is a limitation of Microsoft Windows which is not capable to deal with multiple instances of 16bit applications which have multiple writable data segments. In the past, any try to execute a second dialog server resulted in a WinExec error 16.

In order to overcome this limitation (which would prevent you from starting more than one graphical application simultaneously) a workaround has been implemented in HP Eloquence.

Microsoft Windows recognizes an application by its *module identifier*. A *module identifier* is simply a string at a known location in the executable file holding the internal application name. On application start-up, the Windows kernel checks if an application with this *module identifier* is already running. This has runtime benefits since the code segments of multiple instances are shared in memory.

The workaround to overcome this limitation is implemented by two means:

1 A new **DLGCLONE** utility has been provided to create additional copies of the **DLGSRV** executable, each with a different *module identifier*. These copies are named **DLGSRV2.EXE** up to **DLGSRV9.EXE**.

2 The **RUNSRV** utility now has a built-in mechanism to take care if **DLGSRV** is already active. If it detects an already active **DLGSRV**, it starts one of the additional copies created by **DLGCLONE** instead. This way, multiple instances of **DLGSRV** can be run simultaneously since Microsoft Windows recognizes them as different modules due to different *module identifiers*.

This approach has some drawbacks:

- You need about 1.2 MB of additional disc space for each possible **DLGSRV** instance.
- You need about 1.2 MB of available memory for each running **DLGSRV** instance since the code segments are not shared between multiple instances.

To activate this mechanism, the eloq.ini file on the PC requires the following changes:

1 Replace the *DlgSrv* configuration item in section *[runsrv]* with *DlgSrvX*.
DlgSrvX requires an additional entry of **%s** immediately following the base name of the **DLGSRV** executable file.
If your *DlgSrv* is e.g.:

```
DlgSrv = C:\ELOQ\DLGSRV.EXE -connect %s -IDMfont 1 -IDMcolor 1
```

replace it with:

```
DlgSrvX = C:\ELOQ\DLGSRV%s.EXE -connect %s -IDMfont 1 -IDMcolor 1
```

2 Add the *NDlgSrv* configuration item in section *[runsrv]*. This specifies the maximum number of simultaneously running **DLGSRV** instances. The maximum value for *NDlgSrv* is 9.

If you require e.g. four instances of **DLGSRV** running simultaneously, you specify:

```
NDlgSrv = 4
```

3 After saving your changes to the eloq.ini file, run the **DLGCLONE** utility.

NOTE: Always run the **DLGCLONE** utility after changing the values of *DlgSrvX* and/or *NDlgSrv*. It will automatically adjust the number of copies of the **DLGSRV** executable to the value given in the *NDlgSrv* configuration item. The copies will be created in the directory named by the *DlgSrvX* configuration item.

NOTE: The *DlgSrv* configuration item serves for backward compatibility and may not be supported in future releases anymore. *DlgSrvX* is much more flexible and should be used instead.

NOTE: Please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [runsrv]* for details about *DlgSrvX* and *NDlgSrv*.

The Remote Exec Utility

The **EQEXEC** utility is now provided for the Microsoft Windows platform. It makes it possible to start HP-UX processes on a remote host using the *rexec* protocol.

EQEXEC enables you to start your HP-UX application without the necessity to login to your favorite HP-UX box while you don't get in contact with the operating system at all.

EQEXEC is designed for end users to start their HP Eloquence applications on a PC running Microsoft Windows using the **DLGSRV** network dialog server. This way, HP Eloquence applications behave like native Microsoft Windows programs.

Probably, **EQEXEC** may be used in a more general way since the *rexec* protocol enables you to start any process on any remote host running a remote execution server.

NOTE: For details about the *rexec* protocol please refer to the *rexecd(1m)* HP-UX manual page.

EQEXEC Features

The **EQEXEC** utility

- is fully localizable by changing a configuration file
- supports multiple hosts, applications and users
- provides a command line per application (up to 500 characters)
- has a built-in configuration screen, so it is easy to use and maintain
- includes a log window for debugging purposes

As an additional benefit, if your HP Eloquence application has been migrated to Dialog Manager client-server dialogs, you don't need a HP-UX user license (since the HP-UX system is used as an application server only) and a costly terminal emulator such as Reflection in order to start your application.

The EQEXEC.INI File

The **EQEXEC** utility relies on the eqexec.ini configuration file located in your **WINDOWS** directory. Part of this file is automatically maintained by **EQEXEC**, part of it can be manually configured. The eqexec.ini file is fully documented and contains a complete description of the various configuration items.

Manual configuration includes

- the common runtime behavior (e.g. if a password is required generally)
- customizing the dialog caption and control element titles
- customizing the program error messages

Using EQEXEC

EQEXEC pops up with a dialog. It contains the following control elements:

Application: Select the application you want to execute.

NOTE: The *application* is saved for the next time you use **EQEXEC**.

Username: Enter your user name to log in on the remote host.

NOTE: The *username* is saved for the next time you use **EQEXEC**.

Password: Enter your password to log in on the remote host.

NOTE: The *password* is *not* saved for the next time you use **EQEXEC**.
It depends on the *PasswordRequired* configuration item in section *[common]* of the eqexec.ini configuration file if a password is required generally.

Log output: Check this option to show the host response in a log window. This is useful if a command does not execute as expected.

NOTE: The state of the *log output* check box is saved for the next time you use **EQEXEC**.

OK: Push this button to execute the selected application.

Cancel: Push this button to leave **EQEXEC**.

Settings: Push this button to open the configuration sub-dialog.

The configuration sub-dialog enables you to maintain the application list. It contains the following control elements:

Description: This is the application title to be displayed in the list.

Host: This is the name of the remote host where the application shall be executed.

NOTE: You should provide the appropriate entry in your **HOSTS** file. The location of your **HOSTS** file depends on your networking software.

Command: This is the command line to be executed on the remote host.

NOTE: Is is recommended to redirect *standard input* to the *null device*, this is done by appending **</dev/null** to the command line.
When the command is executed as expected, you should also redirect *standard output* and *standard error* to the *null device* since this reduces the network traffic. This is done by appending **>/dev/null 2>&1** to the command line.

Add: Push this button to put your changes to the application list.

Delete: Push this button to delete the selected entry from the application list.

NOTE: Since every control element title can be customized, the titles in this description may differ from the current **EQEXEC** settings.

On-line Help

The HP Eloquence dialog server provides a mechanism which calls an external program to provide (context sensitive) on-line help.

The HP Eloquence dialog server relies on the popular Netscape WWW browser to provide on-line help. This makes it possible to have the on-line documentation in standard HTML format placed on a server system (running a HTTP server) or in a local directory.

As another benefit, the on-line documentation format is portable across the different systems such as HP-UX and Microsoft Windows which allows easier maintenance of the documentation files.

NOTE: This functionality requires Netscape revision 1.1 or above.

NOTE: The help mechanism built into former HP Eloquence dialog server releases worked different, depending on the dialog server. The motif dialog server called the HP VUE help subsystem (by executing *helpview*). The Microsoft Windows dialog server called the Microsoft Windows help subsystem. This implied different documentation formats from system to system. The help context passed to the external program was created from information contained in the dialog variable *HelpVolume* and an object specific help attribute.

NOTE: Netscape is not included with HP Eloquence. It can be purchased separately from *Netscape Communications Corp*. Netscape can either be obtained from a local distributor or may be downloaded from the anonymous ftp server of *Netscape Communications Corp*. at *ftp.netscape.com*.

The Microsoft Windows Platform

The Microsoft Windows dynamic data exchange (DDE) communication protocol is used to communicate with Netscape.

NOTE: Please refer to section *The RUNSRV Utility, RUNSRV DDE Communication* prior in this chapter for details about Microsoft Windows dynamic data exchange.
Additional information is provided in chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [modules].*

The Motif Platform

HP Eloquence uses the communication mechanism built in Netscape. It calls the Netscape executable passing any arguments on the command line.

Netscape will then pass the request to an already running Netscape process on the current display. If Netscape is not currently running on the display, it will be started. In order to locate the Netscape executable, the dialog server relies on the configuration item *Netscape* in section *[dmsrv]* of the eloq.ini file on the HP-UX system.

NOTE: Please refer to chapter *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the HP-UX System, Section [dmsrv]* for details.

Accessing the On-line Documentation

Before the on-line documentation can be accessed, two configurations are necessary:

1 The configuration item *HelpBaseURL* defines the base location of the on-line documentation.

On the HP-UX system, you define *HelpBaseURL* in section *[dmsrv]* of the eloq.ini file on the HP-UX system.

On the PC platform, you define *HelpBaseURL* in section *[dlgsrv]* of the eloq.ini file on the PC.

Example:

```
HelpBaseURL = http://www/application/help/
```

You can think of *HelpBaseURL* as the on-line help document root for a specific HP Eloquence application. Below this root there should be separate hierarchy levels, one for each functional area inside the application.

NOTE: Please refer to chapters *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the HP-UX System, Section [dmsrv]* and *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [dlgsrv]* for details.

2 The *EqHelpPath* dialog variable defines the on-line document location relative to *HelpBaseURL*. This variable can either be set to a fixed value in your Dialog Manager dialog file, e.g.:

```
config variable string EqHelpPath := "accounting/Dialog.html";
```

Alternatively, it can be dynamically set in your HP Eloquence program using the **DLG SET** statement, e.g.:

```
DLG SET "EqHelpPath!value","accounting/Dialog.html"
```

You can think of *EqHelpPath* as the on-line help document hierarchy level for a specific functional area inside the HP Eloquence application. The objects inside this functional area should be each associated with a separate *help tag* in order to realize a context sensitive on-line help system.

NOTE: Please refer to section *Dialog Manager, Accessing the Dialog Manager Variables* later in this chapter for details.

If these prerequisites are fulfilled, a *help tag* can be defined for every dialog object using the **.help** attribute. This is normally done in your Dialog Manager dialog file, e.g.:

```
window My_window
{
   ...
   .help "#My_window";
   ...
}
```

Alternatively, it can be dynamically set in your HP Eloquence program using the **DLG SET** statement, e.g.:

```
DLG SET "My_window.help","#My_window"
```

With these settings, the final document address is composed by concatenating *HelpBaseURL*, *EqHelpPath* and *My_window.help*. The resulting document is:

```
http://www/application/help/accounting/Dialog.html#My_window
```

Pressing the F1 function key or triggering an *EqRule -1* will then automatically invoke the Netscape WWW browser, which in turn loads the resulting document named above.

NOTE: You can specify a different **`.help`** attribute for each dialog object in order to provide real context sensitive on-line help. However, if an object does not have a **`.help`** attribute, the parent object's **`.help`** attribute is used if present. This way, the object hierarchy is traced up to the root (*window*) object until a **`.help`** attribute is found.

NOTE: If you want to arrange your on-line help document hierarchy in the local file system rather than on a HTTP server, you simply define a different *HelpBaseURL*, e.g.:

```
HelpBaseURL = file://opt/application/help/
```

The DLG HELP Statement

The **`DLG HELP`** statement is now implemented with the dialog server. It provides access to a specific *help tag* as if the F1 function key had been pressed or an *EqRule -1* had been triggered.

Following the example in the former section, if you issue the following statement:

```
DLG HELP "#My_window"
```

Netscape will be invoked loading the same document as in the example above.

The EqHelpOnObject Function

This function is used internally to enable context sensitive on-line help when pressing the F1 function key or triggering an *EqRule -1*.

You can apply this function inside a Dialog Manager rule, e.g.:

```
on WINDOW help
{
   EqHelpOnObject(this);
}
```

It cannot be applied inside a HP Eloquence program, since it expects an *object identifier* argument which HP Eloquence cannot provide.

This function must be declared in your Dialog Manager dialog file if you want to apply it, the declaration should look like:

```
function boolean EqHelpOnObject(object input);
```

Please refer to the file **`/opt/eloquence/lib/defaults.eq`**, where you will find the function declaration as well as the **`on WINDOW help`** rule.

The EqHelpOnTag Function

This function is similar to the **`EqHelpOnObject`** function, except that an additional parameter must be provided which refers to a specific *help tag*.

You can apply this function inside a Dialog Manager rule to provide on-line help for a specific *help tag*, e.g.:

```
on Help_on_help_button select
{
   EqHelpOnTag(this,"#HelpOnHelp");
}
```

It cannot be applied inside a HP Eloquence program, since it expects an *object identifier* parameter which HP Eloquence cannot provide.

This function must be declared in your Dialog Manager dialog file if you want to apply it, the declaration should look like:

```
function boolean EqHelpOnObject(object input, string input);
```

Please refer to the file **`/opt/eloquence/lib/defaults.eq`**, where you will find the function declaration.

The EqHelpViewFile Function

You apply this function to view the contents of any text file using the Netscape WWW browser.

The configuration item *FileBaseURL* defines the base location of the files to be viewed. This is similar to the *HelpBaseURL* configuration item.

On the HP-UX system, you define *FileBaseURL* in section *[dmsrv]* of the eloq.ini file on the HP-UX system.

On the PC platform, you define *FileBaseURL* in section *[dlgsrv]* of the eloq.ini file on the PC.

Example:

```
FileBaseURL = http://www/application/documents/
```

You can think of *FileBaseURL* as the root directory containing any files related to a specific HP Eloquence application.

NOTE: Please refer to chapters *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the HP-UX System, Section [dmsrv]* and *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [dlgsrv]* for details.

The **`EqHelpViewFile`** function expects a *string* argument which is appended to *FileBaseURL* to compose the final document address.

Example:

```
DLG CALL FUNCTION "EqHelpViewFile"("order.lst")
```

This will invoke Netscape, which in turn will load the following document:

```
http://www/application/documents/order.lst
```

This function must be declared in your Dialog Manager dialog file if you want to apply it, the declaration should look like:

```
function boolean EqHelpViewFile(string input);
```

Please refer to the file **`/opt/eloquence/lib/defaults.eq`**, where you will find the function declaration.

NOTE: If you want to arrange your file hierarchy in the local file system rather than on a HTTP server, you simply define a different *FileBaseURL*, e.g.:

```
FileBaseURL = file://opt/application/documents/
```

The EqHelpManPage Function

You apply this function to view the contents of any manual page using the Netscape WWW browser.

The configuration item *ManBaseURL* should be set to the location of a *CGI script* providing the functionality to retrieve a manual page using a search expression.

On the HP-UX system, you define *ManBaseURL* in section *[dmsrv]* of the eloq.ini file on the HP-UX system.

On the PC platform, you define *ManBaseURL* in section *[dlgsrv]* of the eloq.ini file on the PC.

Example:

```
ManBaseURL = http://www/cgi-bin/man2html
```

NOTE: Please refer to chapters *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the HP-UX System, Section [dmsrv]* and *Configuring HP Eloquence*, section *Customize the ELOQ.INI File on the PC Platform, Section [dlgsrv]* for details.

The **`EqHelpManPage`** function expects a *string* argument which is appended to *ManBaseURL* to compose the final man page address.

Example:

```
DLG CALL FUNCTION "EqHelpManPage"("?man")
```

This will invoke Netscape, which in turn will load the following document:

```
http://www/cgi-bin/man2html?man
```

This function must be declared in your Dialog Manager dialog file if you want to apply it, the declaration should look like:

```
function boolean EqHelpManPage(string input);
```

Please refer to the file **`/opt/eloquence/lib/defaults.eq`**, where you will find the function declaration.

The CVDLG Utility

CVDLG is a HP-UX utility which converts HP Eloquence dialog files into Dialog Manager dialog files.

HP Eloquence dialog files, if accessed by a driver, will be converted temporarily into Dialog Manager format each time a dialog is loaded. However, this is a time consuming process and you may also wish to optimize the dialog layout using the Dialog Manager graphical editor. You may use **CVDLG** to convert HP Eloquence dialog files into Dialog Manager dialog files.

Dialog File Name Extensions

The following file name extensions are recommended:

.dlg	HP Eloquence dialog file
.idm	Dialog Manager dialog file
.idc	compiled Dialog Manager dialog file

The default file name extensions may be re-defined using the eloq.ini file.

NOTE: Compiled Dialog Manager dialog files are platform specific.

Using CVDLG

CVDLG is used with the following command line:

cvdlg [*arguments*] outfile infile [infile ...]

outfile: This specifies the destination file name. An extension of **.idm** is recommended.

infile: This lists the HP Eloquence dialog files to be converted. At least one *infile* must be specified.

The optional *arguments* must be supplied in the following order:

1 **-help**
Show program usage.
2 **-debug*n***
Set the internal **CVDLG** debug level to ***n***.
3 **-driver *driver arguments***
Specify the dialog driver and additional driver arguments. This is equivalent to the parameter supplied to the **DLG SET ".driver"** statement. Please refer to section *HP Eloquence Dialog Drivers* prior in this chapter.

Example:

```
-driver "motif myoptions -IDMtracefile /tmp/idmtrace"
```

This will start the motif driver, specifying the *user-defined section* named *[myoptions]* and an additional Dialog Manager argument which creates a trace file for debugging purposes.

NOTE: If **-driver** is not specified, the **DRIVER** environment variable is used instead. This variable should be set to ***driver arguments***, e.g.:

```
DRIVER = "motif myoptions -IDMtracefile /tmp/idmtrace"
```

NOTE: **CVDLG** uses HP Eloquence drivers to convert dialog files.
If using the network driver, the output files will be created on the system running the **DLGSRV** utility. So the destination file name must be specified according to the remote system requirements. If you specify a relative file name, the target file will be created relative to the directory specified in the eloq.ini file.

CVDLG Examples

Example 1:

```
cvdlg -driver "motif myoptions" sample.idm sample.dlg
```

This example will convert a HP Eloquence dialog file named *sample.dlg* into a Dialog Manager dialog file named *sample.idm* using the motif driver. The *user-defined section* named *[myoptions]* is used for driver configuration.

Example 2:

```
cvdlg -driver @client c:\\dlg\\sample.idm c:\\dlg\\sample.dlg
```

This example will convert a HP Eloquence dialog file named *sample.dlg* into a Dialog Manager dialog file named *sample.idm*. The network dialog driver **DLG–SRV** is used to perform this conversion, running on the remote PC named *client*.

Any *backslash* path separator character must be doubled since the shell uses a *backslash* as an escape character. You may use the regular HP-UX *slash* character instead, it will be mapped to *backslash* by the dialog driver.

The Conversion Process

A HP Eloquence dialog file is translated in a sequence of **DLG NEW** and **DLG SET** statements which are executed by the driver process.

The following sequence of HP Eloquence statements reflect the internal operation of **CVDLG**:

```
10 DLG SET ".driver","motif"
20 DLG LOAD "sample.dlg"
```

```
30 DLG CALL RULE "EqSaveDialog","EqDialog"("sample.idm")
40 DLG STOP
```

The following example script will show a possible usage of the **CVDLG** utility using the motif driver. It will convert **.dlg** files to **.idm** files. The dialog files are afterwards post-processed by *sed* (the dialogs are renamed in this step).

If no file names are given on command line, this script will convert all **.dlg** files in the current directory which have no corresponding **.idm** files. If any file name is specified on the command line, the corresponding **.idm** files are overwritten.

```
#! /bin/sh
echo "mkidm.sh"
force=n
defaults=defaults.eq

if [ -z "$DISPLAY" ]
then
   echo "Motif display required!"
   exit 1
fi

if [ $# != 0 ]
then
   list=$*
   force=y
else
   list="*.dlg"
fi

for i in $list
do
   base=`basename $i .dlg`
   idm=$base.idm
   idc=$base.idc

   if [ ! -f $idm -o $force = y ]
   then
      echo "\\n$i -> $idm"

      rm -f $idm $idc

      /usr/eloquence/cvdlg -driver motif +arg $defaults $idm $i
      if [ $? != 0 ]
      then
         echo "failed!!"
         exit
      fi

      sed -e "s/EqDialog/Dlg_$base/g" %<$idm >$$
      if [ $? != 0 ]
      then
         echo "sed failed!!"
         exit
      fi
      mv $$ $idm
   fi
done
```

ASCII DLG and Dialog Manager Compatibility

Although HP Eloquence dialog handling and Dialog Manager are quite compatible (due to mapping performed by the driver), there are some differences.

This document describes the different behavior between HP Eloquence and Dialog Manager.

Object Labels

Object labels in HP Eloquence ASCII DLG are not case sensitive, but they are in Dialog Manager. Dialog manager requires the first character of an object label to be uppercase. HP Eloquence will convert the first character in an object label to uppercase automatically.

Dialog

The HP Eloquence *Dialog* object type is mapped to the Dialog Manager *window* object type.

The Dialog Manager window titles are displayed centered, whereas HP Eloquence places dialog titles near the upper left edge.

The *border* attribute has no functionality in Dialog Manager, except that if *border* is set to zero, no window title will be displayed.

Because modal dialog windows are used to map the HP Eloquence dialog, you must be careful about changing the visual order of windows on the screen. Most GUI systems require closing the top window before another window may be activated again.

PushButton

The following attributes have no functionality in Dialog Manager:

- *.adjust*
- *.border*

If the object height is > 2 then the object height will be reduced by 2.

GroupBox

Dialog Manager *groupbox* objects do not have a title bar.

ListBox

Dialog Manager *listbox* objects do not have a title bar. The HP Eloquence *multi-line* attribute is not supported.

NOTE: Be careful with the *file* attribute: The corresponding file will be read from the local system (where the dialog server is executed). If you are using the network dialog server on the PC platform, this file is expected on the PC. This behavior is subject to change in a subsequent release.

EditText

Dialog Manager *edittext* objects do not have a title bar.

In a graphical environment (Motif, Microsoft Windows), *edittext* always operates in *insert mode*.

The *maxlines* attribute is not supported. The attribute value will always be 0.

NOTE: Be careful with the *file* attribute: The corresponding file will be read from the local system (where the dialog server is executed). If you are using the network dialog server on the PC platform, this file is expected on the PC. This behavior is subject to change in a subsequent release.

The DEFAULTS File

The defaults file is used to provide defaults to dynamically created objects and will be used by the **CVDLG** utility program to convert dialog files. In addition it controls the dynamic behavior of the dialog server.

This chapter will shortly introduce each major element of the HP Eloquence defaults file. Please refer to the *ISA Dialog Manager* documentation for further reference.

Callback Functions

The callback functions are used by HP Eloquence to filter unexpected events. If a unexpected event is recognized, it will be ignored. All further processing is done through the appropriate rules.

HP Eloquence needs the functions below to keep track of the current object state.

Function name	Called
EqPushButtonCB	if a pushbutton is selected
EqCheckBoxCB	if a checkbox is (de-) activated
EqRadioButtonCB	if a radiobutton is activated
EqEditTextCB	if an edittext looses focus

Returning from DLG DO

Returning from the **DLG DO** statement is achieved by calling the procedure **EqExitEventLoop**. It will return control back to HP Eloquence with a given return code and an object path.

Function name	Description
EqExitEventLoop	This will terminate the **DLG DO** statement with the given object and rule value.

Resources

Resources are platform dependent. *Resource variants* allow to have the same dialog source file for all platforms.

Whenever there are resource variants, they are used in the following manner:

Variant	**Platform**
0	Motif GUI
1	Microsoft Windows GUI
2	Alpha Windows

To activate the appropriate resource variant you specify it on the driver command line, e.g.:

```
-IDMfont 1 -IDMcolor 1
```

This will start-up Dialog Manager with font and color variant 1. Please refer to the *ISA Dialog Manager* documentation for a description of valid command line arguments.

The following font resources are allocated by default:

- **`BaseFont`** is used to calculate all sizes and positions.
  BaseFont should be a fixed space font.
- **`NormalFont`** is used to display all text fields

Attributes

There are some user-defined attributes allocated for use by HP Eloquence:

- ***`integer EqRule`***
  This is used to emulate the HP Eloquence *rule* attribute.
- ***`integer EqActiveline`***
  This is used to emulate the HP Eloquence *activeline* attribute.
- ***`integer EqKbRule[10]`***
  This is used to emulate the HP Eloquence function key handling.

Popup Box

The HP Eloquence **`POPUP BOX`** statement is mapped to the **`EqPopup`** window and its associated rules.

It will be configured dynamically by HP Eloquence according to the needs of a **`POPUP BOX`** statement.

Rules

- **`rule boolean EqSaveDialog( string Fname )`**
  This is used by the **`CVDLG`** utility to save the converted dialog files.
- **`on CHECKBOX activate, deactivate`**
  This is used to map *CheckBox* behavior.
- **`on RADIOBUTTON activate`**
  This is used to map *RadioButton* behavior.
- **`on LISTBOX dbselect`**
  This is used to map *ListBox* behavior.
- **`on LISTBOX key EqKbSelect`**
  This is used to map *ListBox* behavior.
- **`on EDITTEXT deselect`**
  This is used to map *EditText* behavior.
- **`on dialog key EqKbF1`** … **`on dialog key EqKbF8`**
  This is used to map HP Eloquence function key handling.

Customizing the Dialog Server

HP Eloquence dialog servers are available as object libraries and partial source code.

This has been provided to allow

- adapting the dialog server to new Dialog Manager releases
- extending the dialog server

Obtaining the Dialog Server Development Files

On the PC platform, you may optionally install the **DLGSRV** development files. They are installed by default in the directory **C:\ELOQ\DLGSRV**.

On the HP-UX system, there is a **driver.tar** archive file in the **/opt/eloquence/share/dist** directory.

Adapting the Dialog Server to New Dialog Manager Releases

Installing a new Dialog Manager release may make it necessary to re-link the dialog server (on the PC platform **DLGSRV.EXE**, on the HP-UX system **motif.drv** and **alpha.drv**) with the new Dialog Manager libraries.

HP-UX System

You need the following software components installed on your HP-UX system:

- ANSI C compiler
- X11/R5 shared libraries
- Motif 1.2 shared libraries
- ISA Dialog Manager Runtime

Install the new Dialog Manager release on the system in the default location which is **/usr/IDM** at the present time.

Extract the files from the **driver.tar** file into any directory. This will create a **driver** subdirectory. Change to the **driver** subdirectory and invoke the linker.

Example:

```
tar -xvf /opt/eloquence/share/dist/driver.tar
cd driver
make
```

This should create a new **motif.drv** and **alpha.drv** file. Install these files in the **/opt/eloquence/lbin** directory.

NOTE: If you did not purchase the optional alpha windows runtime library (**libIDMaw.sl**), the make will fail with a message complaining about a missing library and won't create a new **alpha.drv**. This is not a failure.

PC Platform

You need the following software components installed on your PC:

- Microsoft Visual C/C++ compiler version 1.5 or compatible
- ISA Dialog Manager runtime

Install the new Dialog Manager release on the PC in the default location which is **C:\IDM** at the present time.

Add the Dialog Manager *include* and *lib* directories to your compiler directory setup. These directories are **C:\IDM\INCLUDE** and **C:\IDM\LIB** at the present time.

Change to the directory **C:\ELOQ\DLGSRV** and invoke the linker.

```
c:
cd \eloq\dlgsrv
nmake -f dlgsrv.mak
```

This should create a new **DLGSRV.EXE** file. Install this file in the appropriate directory and run the **DLGCLONE** utility afterwards.

NOTE: You may encounter problems if your compiler is not installed in the directory **C:\VC**. In this case you should edit the *dlgsrv.mak* file and replace the references to **C:\VC**. If you use the Microsoft Visual C/C++ IDE, paths should be fixed automatically when you open the *dlgsrv.mak* project file.

Extending the Dialog Server

Extending the HP Eloquence dialog server makes it possible to move functionality from your application to the dialog server.

This results in

- better application performance
- less network traffic
- better application code structure

The following files are available as source code:

- *bind.c* - declare functions to the Dialog Manager Runtime
- *helpview.c* - interface to the on-line help functionality

NOTE: You should not replace the HP Eloquence dialog server with a modified one without saving a backup of the genuine driver file(s). If you customize the dialog server functionality, you should save a backup copy of the genuine driver file(s) at a secure location before replacing them with the modified driver file(s).

Include Your Own Functions

To include your own functions written in *C language* into the dialog server, you may edit the file *bind.c*. Add your function or procedure definitions and compile/link the dialog server. Now you may call your own function from within the Dialog Manager rules.

For a more specific description please refer to the *ISA Dialog Manager* documentation, chapter *C Application Interface*.

Dialog Manager

Using Models

Dialog Manager models provide an efficient way to handle similar objects instead of defining a lot of individual objects.

Example:

```
model statictext YellowText
{
   .fgc ColBlack;
   .bgc ColYellow;
   .width 10;
   .text "YellowText";
}
...
child YellowText Yt1
{
   .xleft 30;
   .width 15;
   .ytop 0;
}
```

This defines the model *YellowText*. It is derived from the *statictext* object default, but defines its own attribute default values.

The instance *Yt1* of the *YellowText* model will inherit all attributes as defined by the *statictext* default and the *YellowText* model.

The resulting object will display "YellowText" with black text on a yellow background.

Models may also define new attributes for Dialog Manager objects and may have default rules associated with them.

The following example will define the model *MyMenuItem* which is derived from the default *menuitem* object type. *MyMenuItem* defines an *EqRule* attribute and a rule causing return from **DLG DO** statement to HP Eloquence when an instance of the *MyMenuItem* is selected and this instance has a nonzero *EqRule* value:

```
model menuitem MyMenuItem
{
   .text "MyMenuItem";
   integer EqRule := 0;
}
...
child menubox
{
   .title "&File";
```

```
      child MyMenuItem About
      {
         .text "&About";
         .EqRule := 1;
      }
      ...
      child MyMenuItem Exit
      {
         .text "&Exit";
         .EqRule := 2;
      }
   }
   ...
   on MyMenuItem select
   {
      if this.EqRule then
         EqExitEventLoop(this, this.EqRule);
      endif
   }
```

NOTE: Please refer to the *ISA Dialog Manager* documentation for details.

NOTE: When a new dialog object is dynamically created with the **`DLG NEW`** statement (where the *object path* and an *object type* must be provided), you may specify a Dialog Manager object model instead of the *object type*.

Using Modular Dialog Files

Dialog Manager Versions A.03.02a and above allow modular dialog files. This new feature helps to encapsulate reusable information into module files while the dialog file itself is stripped down to its dialog specific information.

NOTE: This feature is currently unsupported. There is a severe bug in the underlying ISA Dialog Manager library (a file descriptor leak), which will cause a fatal dialog server failure after processing a few dialogs.

Although this feature has bugs and is therefore unsupported, we strongly recommend that you get in touch with modular dialog files, because they have great advantages due to reduced redundancies. This will help you to reduce maintenance effort significantly.

The next HP Eloquence patch will provide full support for modular dialog files assuming the Dialog Manager bug has been fixed.

You find a detailed description of modular dialogs in the Dialog Manager release notes 8/95, Version A.03.02a. These release notes will discuss only HP Eloquence specific issues.

There is a new modular HP Eloquence *defaults.eq* file. You find it and the related modules at the following locations:

HP-UX:	**`/opt/eloquence/lib/module`**
Windows:	**`C:\DLG\MODULE`**

This file contains the include directives for the following modules:

eqrsrc	fonts and colors
eqbind	functions, accelerators, standard rules
eqdef	defaults
eqpopup	EqPopup window

You find these files in the module directory named above.

Additionally, there are the related interface files with the extension **`.if`**.

To use these modules, follow these steps:

1 In order to access the module files, you must establish the **`IDMLIB`** access path. This can be done either by using the *IdmLib* configuration item in the eloq.ini file or by setting the **`IDMLIB`** environment variable.

- **HP-UX: eloq.ini, section [dmsrv]**
  IdmLib = /opt/eloquence/lib/module:/opt/eloquence/lib/mymodule
- **HP-UX: `IDMLIB` environment variable**
  IDMLIB = /opt/eloquence/lib/module:/opt/eloquence/lib/mymodule
  export IDMLIB
  (You should include these statements in your *.profile*.)
- **Windows: eloq.ini, section [dlgsrv]**
  IdmLib = C:/DLG/MODULE;C:/DLG/MYMODULE
- **Windows: `IDMLIB` environment variable**
  set IDMLIB = C:/DLG/MODULE;C:/DLG/MYMODULE
  (You should include these statements in your *AUTOEXEC.BAT*.)

The *mymodule* directory is an example for a directory containing your own modules. The directories are separated by a colon ('**`:`**', Motif) or a semicolon ('**`;`**', Microsoft Windows).

2 Adjust the following files according to the contents of your current *defaults.eq* file (if you have never changed the *defaults.eq* please continue with step 4):

- eqrsrc.mod
- eqbind.mod
- eqdef.mod
- eqpopup.mod

3 If you had to change the files in step 2 you should re-create the related interface files.

To do this you change to the module directory named above and execute the **idm** utility with the *+writeexport* option:

```
idm +writeexport interfacefile modulefile
```

Example:

```
idm +writeexport eqdef.if eqdef.mod
```

On Microsoft Windows you should specify the full access path to the **idm** utility and the interface file. You can use the **IDMLIB** value to access the module files (if the **IDMLIB** environment variable has been defined), e.g.:

```
C:\IDM\IDM +writeexport C:\DLG\MODULE\EQDEF.IF IDMLIB:EQDEF.MOD
```

Repeat this for each of the four files in the following order:

- eqrsrc.mod (eqrsrc.if)
- eqbind.mod (eqbind.if)
- eqdef.mod (eqdef.if)
- eqpopup.mod (eqpopup.if)

If you do not have the Dialog Manager development software you should manually adapt the existing interface files:

- Each object defined and exported in a module file has a corresponding declaration statement in the associated interface file.
- So, if you removed exported objects from a module file, you should lookup the corresponding declaration in the interface file and remove it, too.
- If you added new exported objects, you should add a corresponding declaration statement to the interface file.

4 At last you should change your eloq.ini configuration file. You should add a new *user-defined section* which overrides the *DefaultsFile* configuration item to use the new modular one. When starting the dialog server with the **DLG SET** statement, specify this configuration name as an argument, e.g:

```
[UseModules]
DefaultsFile = /opt/eloquence/lib/module/defaults.eq
```

```
DLG SET ".driver","motif UseModules"
```

Alternatively, you can change the *DefaultsFile* setting in the default dialog server section to use the modular *defaults.eq* file:

- **HP-UX:** `DefaultsFile=/opt/eloquence/lib/defaults.eq`
- ***change to*:** `DefaultsFile=/opt/eloquence/lib/module/defaults.eq`
- **Windows:** `DefaultsFile=C:\DLG\DEFAULTS.EQ`
- ***change to*:** `DefaultsFile=C:\DLG\MODULE\DEFAULTS.EQ`

Accessing Dialog Manager Variables

Dialog Manager variables may be set using the **DLG SET** and retrieved using the **DLG GET** statement. The dialog name, the variable name and the *.value* attribute must be specified.

Example:

```
dialog YourDialog
...
variable string StrVal;
variable integer IntVal;
```

You may access these variables from HP Eloquence using the following statements:

```
DLG GET "YourDialog.StrVal!value",A$
```

This will retrieve the value of the Dialog Manager variable *StrVal* into the HP Eloquence variable *A$*.

```
DLG SET "YourDialog.IntVal!value",123
```

This will set the Dialog Manager variable *IntVal* to the constant value *123*.

NOTE: You should use the exclamation mark ('**!**') to delimit variable path and attribute because *native* Dialog Manager attributes are accessed. Please refer to *Accessing Dialog Manager Objects* later in this chapter.

NOTE: Please refer to the *ISA Dialog Manager* documentation for details.

Dialog Manager Image Button

The Dialog Manager *image button* object type is supported with HP Eloquence. Its usage is according to the *pushbutton* object type.

Dialog Manager Records

HP Eloquence supports access to Dialog Manager *record objects*.

Dialog manager record objects are the equivalent of *structures* in *C language* and may be defined globally to the dialog or locally to any object.

This is an efficient method to exchange multiple data with Dialog Manager at once. Using the HP Eloquence **XPACK** and **XUNPACK** statements, you may link HP Eloquence variables to Dialog Manager record members.

Dialog Manager record members may either be defined as shadow objects, such that each access to a record member will affect the associated object attribute, or they may contain data which may be used in Dialog Manager rules or functions.

The **DLG SET** and **DLG GET** statement mappings of the dialog driver have been enhanced to support Dialog Manager record objects.

The DLG SET Statement

The **DLG SET** statement may be used to transfer a buffer, packed by the **XPACK** statement, into equivalent record members of a Dialog Manager record object.

```
DLG SET "Record.@",Buf$
```

NOTE: The .@ attribute must be specified.

Example:

```
window CusWin
{
   ...
   child record CusRec
   {
      string Cus_no shadows CusWin.Cus_number.content;
      string Cus_name shadows CusWin.Cus_name.content;
      boolean Cus_flag shadows CusWin.active;
      ...
   }
}

Cus_no$="1234"
Cus_name$="Customer Name"
Cus_flag=1
XPACK Buf$ FROM Cus_no$,Cus_name$,Cus_flag
DLG SET "CusWin.CusRec.@",Buf$
```

This will transfer the HP Eloquence variables *Cus_no$, Cus_name$* and *Cus_flag* into the Dialog Manager record object *CusWin.CusRec*.

The DLG GET Statement

The **DLG GET** statement may be used to transfer the contents of a Dialog Manager record object into HP Eloquence program variables.

If a .@ attribute is specified, all record members are transferred. If a member name is specified, only the given record member is transferred:

```
DLG GET "Record.@",Buf$
```

This transfers all members of *Record* into *Buf$*.

```
DLG GET "Record.Array",Buf$
```

This transfers all elements of *Record* member *Array* into *Buf$*.

```
DLG GET "Record.Array[1]",Buf$
```

This transfers the first element of *Record* member *Array* into *Buf$*.

Example:

```
DLG GET "CusWin.CusRec.@",Buf$
XUNPACK Buf$
```

This will transfer all members of the Dialog Manager record object *CusWin.CusRec* into equivalent HP Eloquence program variables.

```
DLG GET "CusWin.CusRec.Cus_flag",Buf$
XUNPACK Buf$
```

This will transfer the field *CusWin.CusRec.Cus_flag* into the HP Eloquence program variable *Cus_flag*.

Programming Considerations

In order to use Dialog Manager record objects, the following considerations should be followed:

- The Dialog Manager does not support floating point variables.
- All record member names must be valid HP Eloquence variable names.
- The Dialog Manager record members must have the appropriate type. You may use the Dialog Manager *boolean*, *integer* and *string* data types.
- The Dialog Manager record members and HP Eloquence program variables should have an equivalent type.
- While it is possible to transfer HP Eloquence numeric variables into Dialog Manager *string* record members, the opposite is not possible.

  However, there is a workaround implemented into HP Eloquence which allows to overcome this limitation:

  A Dialog Manager record member with a trailing '**N**' (uppercase) appended to its name is considered numeric, regardless of its real data type. The corresponding HP Eloquence variable name does *not* include the trailing '**N**'. This way, numeric data can be transferred from any Dialog Manager record member data type into HP Eloquence numeric variables.

  Example:

  ```
  window MyWin
  {
     child record MyRec
     {
        string Num_strN "123";
     }
  }


  INTEGER Num_str
  DLG GET "MyWin.MyRec.@",Buf$
  XUNPACK Buf$
  ```

This will transfer the numeric value *123* from the Dialog Manager record member *MyWin.MyRec.Num_strN* (*string* data type) into the HP Eloquence variable *Num_str* (*INTEGER* data type).

NOTE: If such Dialog Manager record members contain non-numeric text, *0* (zero) is transferred into the corresponding HP Eloquence variables.

Dialog Manager Rules and Functions

HP Eloquence provides two statements which enable you to call Dialog Manager *rules* and *functions*.

Rules

Dialog Manager *rules* are implemented using the Dialog Manager script language and are stored in the dialog file. This makes it possible to provide a task-oriented interface to dialogs without having to care about the bits and pieces.

By writing your own rules you actually extend the dialog server functionality. The **`DLG CALL RULE`** statement allows you to trigger such extensions:

```
DLG CALL RULE "Rule","Object" [(arg,arg ...)] [,Retvar[$]] [;Err]
```

Example:

```
window MyWindow
{
   ...
   child image MyImage
   {
      ...
      .text "Image Name";
      .picture Default;
   }
   ...
}
...
rule void LoadGif( string Gif input, string Name input )
{
   this.picture := Gif;
   this.text := Name;
}

DLG CALL RULE "LoadGif","MyImage"("sample.gif","Sample Image")
```

This will trigger the **`LoadGif()`** rule. The object which the rule is applied to (*this*) is *MyImage*, the GIF image to be loaded is *sample.gif* and the image title is set to *“Sample Image”*.

The **`DLG CALL RULE`** statement is mapped to a **`DM_CallRule()`** Dialog Manager function call.

If a return variable is present, the value returned by the rule will be assigned to it. If no return variable is specified, the return value will be ignored.

If the error return variable is present, no runtime error is returned, but the error variable is set with the error number.

NOTE: In the example above, the GIF file will be read from the local system (where the dialog server is executed). If you are using the network dialog server on the PC platform, this file is expected on the PC. This behavior is subject to change in a subsequent release.

Functions

Dialog Manager *functions* are implemented in *C language* and are linked to the dialog server executable. The functions are bound to the Dialog Manager using the **`DM_BindFunctions()`** Dialog Manager function. In order to use such functions they must be declared in the dialog file.

By writing your own functions you actually extend the dialog server functionality. The **`DLG CALL FUNCTION`** statement allows you to trigger such extensions:

```
DLG CALL FUNCTION "Function" [(arg,arg ...)] [,Retvar[$]] [;Err]
```

Example:

```
function boolean EqHelpViewFile(string input);

DLG CALL FUNCTION "HelpViewFile"("sample.txt")
```

This will trigger the **`HelpViewFile()`** function. The file to be viewed is *sample.txt*.

The **`DLG CALL FUNCTION`** statement is mapped to a **`DM_CallFunction()`** Dialog Manager function call. The maximum number of arguments is 8.

If a return variable is present, the value returned by the rule will be assigned to it. If no return variable is specified, the return value will be ignored.

If the error return variable is present, no runtime error is returned, but the error variable is set with the error number. If no dialog server is active, an error 1004 is returned.

NOTE: Please refer to section *Customizing the Dialog Server* prior in this chapter for details about extending the dialog server.

Accessing Dialog Manager Objects

HP Eloquence only knows about a subset of Dialog Manager objects and attributes as necessary to translate HP Eloquence DLG to Dialog Manager.

In order to enable access to Dialog Manager objects and attributes not available in HP Eloquence DLG there is a "bypass option" implemented in HP Eloquence.

If you use an exclamation mark ('`!`') instead of a dot ('`.`') as a separator between object path and attribute, no mapping will be performed. Instead, the *native* Dialog Manager objects and attributes are accessed.

Two examples of *native* Dialog Manager objects are *menubar* and *menuitem*. An example of a *native* Dialog Manager attribute is the *format* attribute of *edittext* objects:

```
DLG SET "MyWindow.Edit!format","%8'3,02sf/1"
```

This will set the field format of the *edittext* object named *Edit*.

Programming Notes

This section covers solutions to common problems concerning Dialog Manager programming issues.

Avoiding Edit Text Exit Rule

Problem: **`on EDITTEXT deselect`** is triggered at **`PUSHBUTTON select`**.

If a *pushbutton* or *image* button is selected, this will involve a focus change which will trigger an **`EDITTEXT deselect`** rule. As a result, it is required to select the *pushbutton* twice.

This behavior can also result in a situation where a "Cancel" *pushbutton* would be unusable if an **`EDITTEXT deselect`** rule performs a field validation and re-sets the focus on itself again.

This problem can be solved in the following manner:

1 Initialize an *integer* variable in your *pushbutton* object or in the object model or default. Do this in every object (model, default) that should have "Cancel" *pushbutton* characteristics, e.g.:

```
model pushbutton Cancel_btn
{
    ...
```

```
      integer EqRuleOverride := 1;
   }
```

2 Change the object (model, default) rule that normally is triggered **`on EDITTEXT deselect`**.

If this rule is e.g.:

```
on EDITTEXT deselect
{
   if this.EqRule then
      EqExitEventLoop(this, this.EqRule);
   endif
}
```

Change this rule to:

```
on EDITTEXT deselect
{
   variable integer RuleOverride := 0;
   if (typeof(this.window.focus.EqRuleOverride)=integer) then
      RuleOverride := this.window.focus.EqRuleOverride;
   endif
   if RuleOverride=0 then
      if this.EqRule then
         EqExitEventLoop(this, this.EqRule);
      endif
   endif
}
```

The basic idea is to obtain the newly focused object (*this.window.focus* which might be the “Cancel” *pushbutton* object) and check if this object has an *EqRuleOverride* attribute. If this is true and *EqRuleOverride* is set the **`EqExitEventLoop()`** function will not be triggered and the **`EDITTEXT deselect`** rule will not perform any action.

Setting the Focus to Microsoft Windows Radio Button Object

Previously, a **`DLG SET "Radiobutton.focus",1`** statement always *activated* the chosen *radiobutton* when applied with the Microsoft Windows network dialog server. This behavior sometimes had side effects if a rule was associated with the activation of this *radiobutton* and the *radiobutton* had not previously been selected.

The current implementation sets the focus to the *currently selected radiobutton within the same group* whenever a **`DLG SET`** statement is used as shown above. Since the focused *radiobutton* has been selected previously, any activation rule will *not* be triggered.

To achieve the former behavior, you can access the *native* Dialog Manager attribute, e.g. **`DLG SET "Radiobutton!focus",1`**

A

Changes Since Release A.05.00

This chapter is a summary of all changes since HP Eloquence release A.05.00.

Release A.05.00

New Functionality

- Integration of a graphical user interface
- COMMAND statement enhancements
- Access to HP Eloquence databases from user-written Microsoft Windows applications
- New ASSIGN mode
- 132 columns on 700/60 terminal
- Alternate definition of the BACKSPACE key
- HP Eloquence DLG enhancements
- FRM2DLG conversion utility

Release A.05.01

New Functionality

- ❐ XPACK, XUNPACK statements
- ❐ DBMODS input redirection
- ❐ DBIMPORT, DBEXPORT field separator
- ❐ DBIMPORT, DBEXPORT performance enhancement
- ❐ Maximum number of open files
- ❐ HOP key
- ❐ Program file size limit
- ❐ LASTLINE command
- ❐ HP Eloquence keyboard handling
    - Typeahead
    - TYPEAHEAD CLEAR
- ❐ Keyboard mapping
    - ON KEYBD #n
    - OFF KEYBD #
    - LASTKEY
- ❐ Store utility enhancements
- ❐ HP Eloquence DLG enhancements
    - Topitem attribute
    - Index line attribute
    - Modification of support of the alt attribute
    - DLG NEW enhancement allows models
    - DLG LOAD enhancements
    - Support of ’include "file"’ in the dialog file
    - Support of the define directive
- ❐ Dialog Manager enhancement
    - Dialog Manager models
    - Dialog Manager image button
    - Dialog Manager records
    - Dialog Manager functions and rules
    - Accessing Dialog Manager objects

Known Problems Fixed

- ❐ Editing a form with more than 128 lines in MFORM could cause all virtual memory used up by HP Eloquence. This usually caused a core dump and could even affect other applications.
- ❐ Returning from a SUB or function caused a DISABLE condition ignored, if a pending ON INPUT interrupt request has been encountered for the parent environment while executing the subprogram.
- ❐ Calling a function or subprogram with improper arguments could cause a core dump (assertion failed: ssp->ofs < seg->symsz). If interrupted at a specific stage, the wrong symbol table became restored.
- ❐ DBINFO mode 202 on a remote database failed when executed on a database other than the first opened one.
- ❐ CRTCOLS= terminates curses subsystem in order to re-program the terminal hardware. When a dialog window was active and curses becomes restarted all windows must be re-created. This could cause a core dump.

  This was caused by a bug in the dialog refresh routine. When only one dialog window was active, the dialog may become destroyed by screen refresh.
- ❐ When a POPUP BOX is executed with Dialog Manager the following problems have been encountered:
  - The first line may not be visible.
  - The default button of the POPUP BOX does not have the focus.
- ❐ The default d.dbstore and d.dbrestore script files did not work.
- ❐ Schema aborted with core dump when a duplicate automatic data set name occurred in the schema description.
- ❐ DBFIND on index item does not work.

  This is a problem in the conversion of the lookup value. PA-RISC and PC systems have different character sets and have a different order of binary values. So each value must be converted before it is understood by the remote system.
- ❐ PC applications calling the ELOQDB.DLL aborted with a GPF if not compiled in large memory model.

  This was caused by a insufficient declaration in ELOQDB.H where status was not declared as FAR.

Release A.05.02

New Functionality

- TYPEAHEAD CLEAR will no longer beep if typeahead data is discarded.
- The DLG DO and POPUP BOX statements are treated like an INPUT statement. The TSTAT is changed to 1 and the time will not account as execution time.
- Support of HPUX 10.x
- The performance of the Windows dialog server has been improved about ten times.
- The Windows dialog server event filter has been improved.
- New "do" attribute for window objects
- Multiple Windows dialog server configurations

Known Problems Fixed

- XPACK / XUNPACK caused a core dump when using REMOTE LISTS
- XUNPACK filter failed to handle array elements properly.
- If the alt attribute is undefined for an HP Eloquence dialog, the '&' will no longer cause the next character to be displayed underlined, because "Keyboard Accelerators" are inactive. This may be important if you use a StaticText object to output data.

NOTE: This behavior is not portable to Dialog Manager since Dialog Manager always enables Keyboard Accelerators.

- Cvdlg will now convert dlg files containing the alt attribute and includes.
- When cloning an HP Eloquence dialog, some dialog specific attributes were not copied: alt, cr, f1 .. f9. A dialog is cloned using the DLG NEW statement, giving an object path instead of an object class.

  Example:

  ```
  DLG NEW "NewDialog","TemplateDialog"
  ```

  will copy the properties of TemplateDialog along with all child objects into NewDialog.

NOTE: Cloning is a recursive operation.

- The list and sfarpt commandline utilities failed with the error message "Inconsistent segment header" if the sum of the code size and the symbol table size of a program segments exceeded 64 kbytes.

- The LOAD and LOAD SUB statements failed with the error 999 "Program file not compatible" if the sum of the code size and the symbol table size of a program segment exceeded 64 kbytes.
- Dbimport internal buffer space has been enlarged. When importing a data set which has an entry length close to the database limit of four kilobytes, an internal buffer overflow was possible.
- Requesting a port outside the range of 11 to 20 caused an internal error if leaving HP Eloquence and the port has not been released.

  Example:

```
REQUEST 99
QUIT
assertion failed: (port >= 11 && port <= 20 && type == 0)
```

- The range of valid PORT numbers have been extended to the PRINTER range. However, only PORT numbers between 11 and 20 are able to process input.
- PRINTER IS 8 (deassigning a printer) could result in the unlock of a requested port.

  Example:

```
REQUEST Port          ! port is locked
PRINTER IS Port       ! port is still locked
PRINTER IS 8          ! port is unlocked
```

- Due to an overflow of an internal Dialog Manager queue, it was possible that the content of (multiline) EditText or ListBox objects and (shadowed) Record elements were not displayed correctly.
- Bad Client State: If a Microsoft Windows Dialog Server was terminated, the async mode was not reset. This could result in internal protocol failures.
- DLG GET ".driver" could result in a random return value.

Release A.05.03

Known Problems Fixed

- ❒ Database support of Journal File System

Release A.05.10

New Functionality

- ❒ A workaround has been implemented to overcome a Microsoft Windows limitation which makes it impossible to have more than one running instance of the HP Eloquence dialog server.
- ❒ The HP Eloquence A.05.10 dialog server is built using the Dialog Manager library revision A.03.01a (on HP-UX) and A.03.02a (on Microsoft Windows).
- ❒ The RUNSRV utility supports Microsoft Windows DDE communication.
- ❒ The dialog server on-line help mechanism has been changed to use the popular Netscape WWW browser.
- ❒ The DLG HELP statement is now implemented for the dialog server.
- ❒ Uniform driver parameters
- ❒ A mechanism has been included in the HP Eloquence dialog server defaults file to enable selection of a PushButton without triggering the exit rule of an EditText.
- ❒ DLG file location controlled by eloq.ini settings
- ❒ The new Dialog Manager modules feature is now available as an UNSUPPORTED feature.

Known Problems Fixed

- ❒ Memory leak: A problem in the ISA Dialog Manager library caused a memory leak when a dialog was loaded and deleted afterwards.
- ❒ Setting the focus on a RadioButton using DLG SET will activate the RadioButton on Microsoft Windows.

B

New Files Since Release A.05.00

This chapter covers the new HP Eloquence file layout and a description of the HP Eloquence files which were added since release A.05.00.

New File Layout Since Release A.05.02

With the release of HP-UX 10.0, Hewlett-Packard will be introducing a new file-system layout paradigm, modelled after SVR4 and OSF. The model provides many benefits, such as separating the operating system from applications and aligning HP with an industry-accepted file system layout.

HP Eloquence follows the HP-UX 10.0 file system conventions and provides a similar file structure with HP-UX 9.x and HP-UX 10.x.

```
/opt/eloquence/                     Base directory
  |
  |-- bin/                          Executable programs
  |
  |-- dll/                          Default DLL location
  |
  |-- include/                      C include files
  |
  |-- lbin/                         Backends to other commands
  |
  |-- lib/                          Object code and libraries
  |    |
  |    |-- nls/
  |    |    |
  |    |    |-- C/                  Default message Catalog
  |    |    |
  |    |    |-- collate/            Files containing language
  |    |                            specific collating sequences
  |    |-- module/                  Modular dialog files
  |
  |-- newconfig/
  |    |
  |    |-- ReleaseNotes/            Release Notes
  |    |
  |    |-- config/                  Example Configuration files
  |    |
  |    |-- startup/                 HP-UX 10.0 start-up/shutdown
  |                                 files
  |-- share/
  |    |
  |    |-- contrib/                 Contributed files
  |    |
  |    |-- dbmods/                  DBMODS related files
  |    |
  |    |-- dist/                    Distributed files
  |    |
  |    |-- example/                 Example files (DVP only)
  |    |
  |    |-- prog/                    HP ELOQUENCE programs
  |    |
  |    |-- terminal/                Terminfo description and
  |    |                            character set mapping
  |    |-- terminfo/                Terminfo database
  |
  |-- tools/                        HP260 migration tools
```

The following directories are only present with HP-UX 9.x:

```
/opt/eloquence/
    |                                  Application specific
    | —— etc/                          configuration files
```

NOTE: With HP-UX 10.x, this directory still contains the license file while the other configuration files are moved to the **/etc/opt/eloquence** directory.

The following directories are only present with HP-UX 10.x:

```
/etc/opt/
    |                                  Application specific
    | —— eloquence/                    configuration files

/var/opt/
    |                                  Application specific
    | —— eloquence/                    temporary files

/etc/
    |
    | —— rc.config.d/                  Start-up configuration files

/sbin/
    |
    | —— init.d/                       Start-up and shutdown scripts
    |
    | —— rc1.d/                        Start-up and shutdown link files
    | —— rc2.d/                        for script sequencing
```

Description of New Files Since Release A.05.00

```
/opt/eloquence/
    |
    |—— bin/
    |    |
    |    |—— chklic                 License file check utility
    |    |—— dbutil                 Database maintenance utility
    |    |—— cvdlg                  Dialog conversion utility
    |    |—— eloqnd                 Remote database daemon process
    |
    |—— lbin/
    |    |
    |    |—— motif.drv              Motif dialog server
    |    |                          (requires /usr/lib/libIDM.sl)
    |    |—— alpha.drv              Alpha windows dialog server
    |                               (requires /usr/lib/libIDMaw.sl,
    |                               not included with HP Eloquence)
    |
    |—— lib/
    |    |
    |    |—— dbutil.dlg             Dialog file for ../bin/dbutil
    |    |—— libIDM.sl              Dialog Manager library for Motif
    |    |—— defaults.eq            Dialog server defaults
    |    |
    |    |—— module/
    |    |    |
    |    |    |—— defaults.eq       Modular dialog server defaults
    |    |    |—— eqbind.mod        Bindings module
    |    |    |—— eqrsrc.mod        Resources module
    |    |    |—— eqdef.mod         Definitions module
    |    |    |—— eqpopup.mod       Popup box module
    |    |    |—— eqbind.if         Module interface files
    |    |    |—— eqrsrc.if
    |    |    |—— eqdef.if
    |    |    |—— eqpopup.if
    |    |
    |    |—— nls/
    |
    |—— share/
         |
         |—— dist/
              |
              |—— pcclnt.exe        Microsoft Windows dialog server
              |                     (self-extracting DOS archive)
              |—— driver.tar        Files required for Dialog server
                                    customizing (tar archive)
```

The following files are new with HP-UX 9.x:

```
/opt/eloquence/
    |
    |—— etc/
         |
         |—— eloq.ini               Dialog server config file
```

The following files are new with HP-UX 10.x:

```
/etc/opt/
  |
  |-- eloquence/
        |
        |-- eloq.ini              Dialog server config file
```

The following files are contained in the **driver.tar** archive:

```
/installation directory/
  |
  |-- driver/                     Dialog server development files
        |
        |-- makefile              Common makefile
        |-- make.driver           Common make definitions
        |
        |-- motif/                Motif files
        |     |
        |     |-- makefile        Makefile for motif driver
        |     |-- bind.c          Bind functions to Dialog Manager
        |     |-- helpview.c      On-line help subsystem
        |
        |-- alpha/                Alpha windows files
        |     |
        |     |-- makefile        Makefile for alpha driver
        |     |-- bind.c          Bind functions to Dialog Manager
        |     |-- no-help.c       On-line help substitute file
        |                         (on-line help is not supported)
        |
        |-- h/                    Header files
        |-- lib/                  Object files and libraries
```

The following files are contained in the **pcclnt.exe** archive:

```
/installation directory/
  |
  |-- runsrv.exe                  RUNSRV utility
  |-- dlgsrv.exe                  Microsoft Windows dialog server
  |-- dlgclone.exe                DLGCLONE utility
  |-- eqexec.exe                  Remote exec utility
  |-- eloqdb.dll                  Remote database DLL
  |
  |-- eloqdb/                     Remote DB development files
  |     |
  |     |-- eloqdb.h              Header file
  |     |-- eloqdb.lib            Import library
  |     |-- eloqdb.def            Definition file
  |
  |-- dlgsrv/                     Dialog server development files
        |
        |-- dlgsrv.mak            Makefile (MS Visual C/C++ 1.5)
        |
        |-- src/                  C source files
        |     |
        |     |-- bind.c          Bind functions to Dialog Manager
        |     |-- helpview.c      On-line help subsystem
        |
        |-- h/                    Header files
        |-- lib/                  Object files and libraries
```