System Administrator's Guide - System Overview

Version 23

Table of Contents

1 System Architecture Overview

ALEPH is a library services agent, providing application services to clients via its APIs (Application Program Interfaces). ALEPH's architecture is based on a multi-tier, client/server model. Client/Server communication is based on a stateless (self-contained) transaction model. Nonetheless, ALEPH Application Servers keep continuous connections (with time-out) to the database, to ensure high performance.

ALEPH features a flexible database design. Each ALEPH site is composed of seven interrelated, yet separate, units: Authority, Bibliographic, Holdings, Administrative, Interlibrary Loan, Course Reading and a system-wide administration unit. Each database unit can have many occurrences with many-to-many links to the other database units. ALEPH's database design supports a wide range of database configurations and implementations, including independent installations of different units on different hosts. ALEPH's database design plays a crucial role in the system's scaleability potential.

ALEPH's architecture is based on a scaleable, distributed logic model and relies on an object-oriented design. Two key features of ALEPH's architecture are:

Multi-tier, Client/Server model - ALEPH is split into logical segments with a clearly defined interface based on message passing. See *Figure 1* below.

Modularity - The key notion of the distributed logic design that underlies ALEPH is modularity - both vertical (between tiers) and horizontal (within the tiers). This ensures that the system is easy to maintain and extend, and that new technologies and concepts can be quickly integrated.

ALEPH can be installed on the following platforms:

- SUN SOLARIS
- LINUX REDHAT

1.1 ALEPH Server Architecture

Multitier Architecture - OPAC

Presentation Services & Logic	GUI Client	WWW	Terminal	Z39.50 Client

The figure below illustrates the ALEPH multitier architecture:
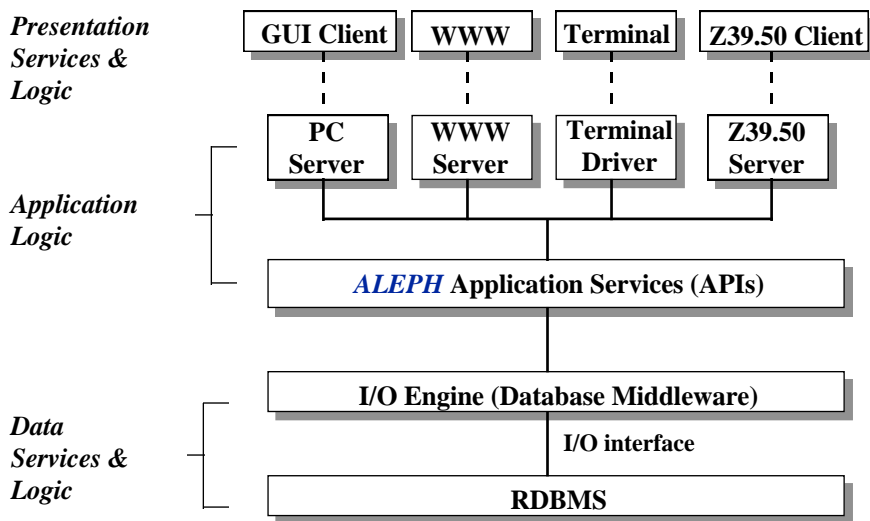
Figure 1

The ALEPH server is composed of the following tiers:

1.1.1 Presentation Services & Logic

Provides the interface with which the user interacts.

1.1.2 Application Logic

Application Servers tier - A front-end tier which is composed of dedicated servers for each interface. Each application server receives a query from a source client, translates the query to a uniform format and directs it to the relevant Application Service object (API). Once translated, all queries have the same format, regardless of their origin (Z39.50 client, ALEPH client, Web browser or standard character mode terminal).

Application Services tier (APIs) - The heart of ALEPH is the application services tier, composed of sets of APIs. The APIs provide library services for the different clients. For example, a FIND API provides FIND services to all clients after a FIND query is invoked at one of the clients (WWW, Z39.50, Telnet or ALEPH proprietary). As part of its open system architecture, ALEPH includes mechanisms to integrate new APIs as well as to extend the Application Services to other clients or applications. This provides considerable potential for extensibility.

1.1.3 Data Services and Logic

ALEPH Database Middleware (or I/O Engine) - This is a high level database management layer. A logical server provides data services to the application services objects. It contains a group of objects which intermediates between the application and the database. The I/O engine translates an application request to a sequence of database commands. In addition, the I/O engine provides SQL enhancement which is required because of the textual, non-formatted nature of library catalog data.

ALEPH's I/O engine also exploits the knowledge that the system has about the DBMS's special characteristics, in order to optimize data updating and retrieval. It is in this logical layer that ALEPH incorporates its experience and know-how of libraries' special data structures and formats.

Having an intermediate level of the I/O Engine between the application and the DBMS ensures maximum flexibility of DBMS logical and physical design.

ALEPH Database - The ALEPH database runs under Oracle RDBMS.

1.1.4 Other characteristics of ALEPH's architecture:

Network Orientation - ALEPH's distributed logic is designed to fit into scaleable network configurations. With its Application Servers tier and Database Middleware tier (the I/O engine), ALEPH is suitable for intra-networking and inter-networking. ALEPH not only supports a range of clients access (WWW, Z39.50, ALEPH proprietary and Telnet clients) but also accommodates access to heterogeneous database resources.

Special features are included to overcome the diversity of databases: MATCH to locate remote/external documents, and Multi-FIND to broadcast a search across multiple databases. In addition, ALEPH includes enhanced functionality to support centralized, de-centralized and union catalogs.

Scalability - ALEPH's multi-tier, distributed architecture provides a wide range of scalability possibilities: distribution of data across disks/servers, distribution of services across servers or even a multi-server configuration with shared data.

2 ALEPH Unix Logins

The ALEPH server requires the following Unix users:

Login	Home Directory	Description
aleph	/exlibris/aleph	ALEPH administrator
oracle	$ORACLE_HOME	Oracle administrator (DBA)

The person who is in charge of all the libraries, the System Administrator, can log in as aleph, and thus will be able to modify the parameters and data of all libraries.

aleph – The ALEPH administrator (System Administrator and/or System Librarian) uses the **aleph** account for various online utilities and command line activity. The **aleph** user has access and control over all the libraries in the system.

oracle – The oracle DBA uses the oracle account for DBA activity outside the scope of online utilities **UTIL O - Oracle Management** and **UTIL A** - **File Administration and Building**.

3 ALEPH Directory Structure

3.1 The aleph Root Directory
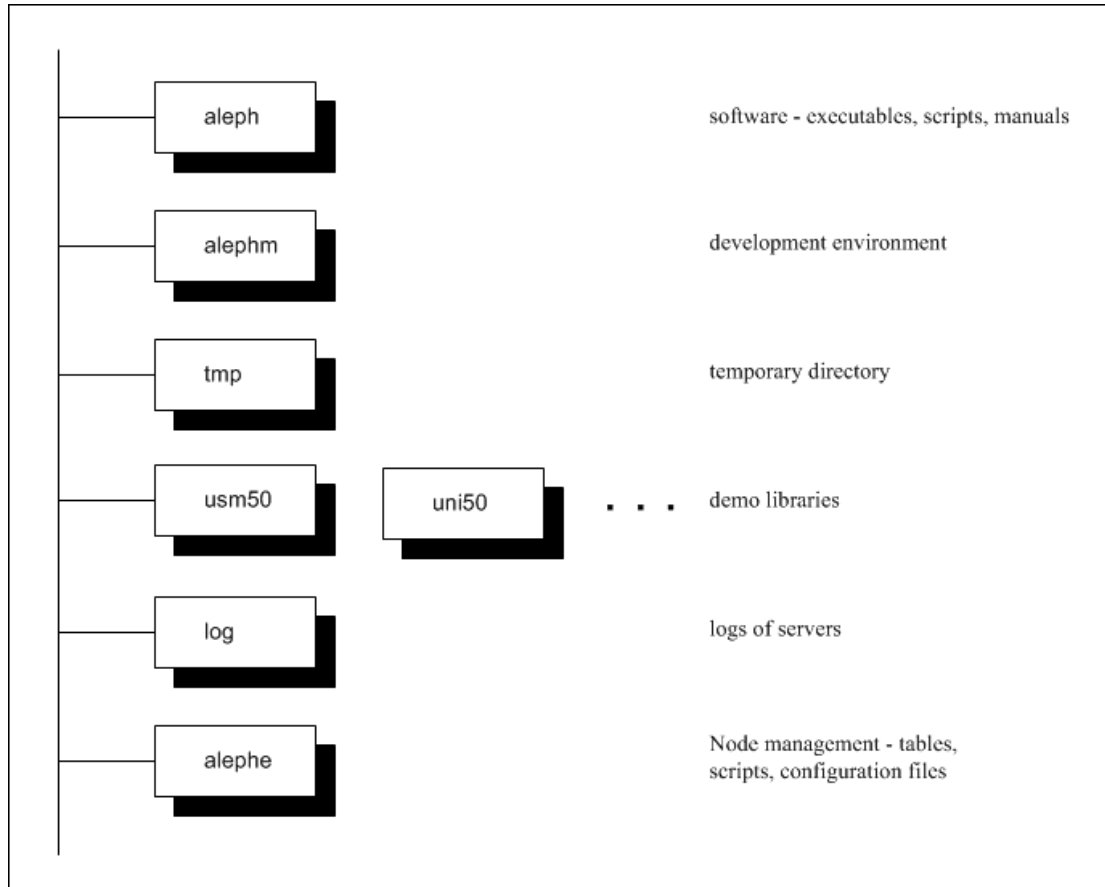
Figure 2

An aleph root is composed of three types of directories:

1. ALEPH software	`alephm, aleph, tmp, log`

2. Demo libraries	`usm01, usm10, usm20, usm30, usm50, usm60`
	`uni01, uni10, uni20, uni30, uni50, uni60`
	(`usm` = MARC21, `uni`=UNIMARC)

3. Node management	`alephe`

As part of the installation phase, the `alephe` directory (node management) is relocated to a different directory (for example `u18_1`). This directory will normally contain site-specific libraries as well. A sample scheme is given in Figure 3.

ExLibris Aleph
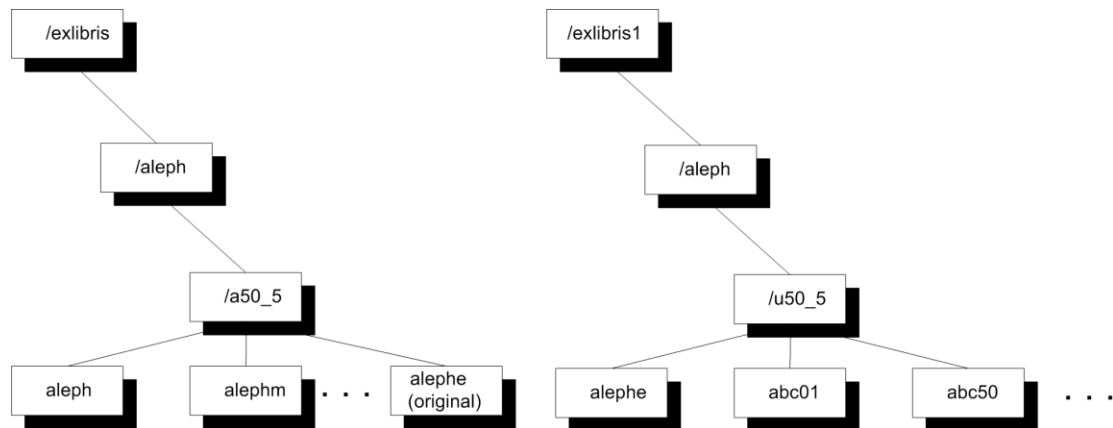
Figure 3

This structure supports the ability to upgrade the ALEPH software (under `a18_1`) without interfering with local customization (under `u18_1`).

In addition to the core software directories `aleph` and `alephm`, the system has directories which include management tables and parameter files pertaining to the specific installation (`alephe`), and to each of the libraries.

3.2 The alephe Directory

The primary directories in the `alephe` tree are:

`tab`	node management tables
`www_<type>_<lng>`	HTML files for Web OPAC, Web services, Course Reading and others
`pc_b_<lng>`	GUI services
`scratch`	intermediate files and logs
`apache`	`conf/htdocs/logs`
`error_<lng>`	messages that are displayed to the end user
`gate`	Z39.50 setup
`unicode`	tables which translate from and to Unicode

3.2.1 The Main Configuration File - aleph_start

`aleph_start` is ALEPH 500's main configuration file. It contains definitions of ALEPH libraries (`abc01`, `usm01`, etc.) and environment variables. It also contains logical assignments (the only place with physical references to ALEPH directories). `aleph_start` is in the `alephe` directory (`cd $aleph_root`).

In order for changes in `aleph_start` to take effect, you must exit ALEPH, re-login, and then restart the daemons and servers. Running `aleph_shutdown` stops all daemons and servers. Running `aleph_startup` restarts daemons and servers depending on definitions in `aleph_start` and `aleph_start.private`.

Following are primary portions of `aleph_start`:

Version information
```
setenv     ALEPH_VERSION        18
setenv     ALEPH_COPY           1
setenv     ALEPH_APP_VERSION    a${ALEPH_VERSION}_${ALEPH_COPY}
setenv     ALEPH_REVISION       01
```

Port definitions
```
setenv     WWW_SERVER_PORT    499${ALEPH_SUB_PORT}
setenv     PC_SERVER_PORT     699${ALEPH_SUB_PORT}
setenv     Z39_GATE_PORT      799${ALEPH_SUB_PORT}
setenv     HTTPD_PORT         8991
setenv     PDS_PORT           8991
setenv     Z39_SERVER_PORT    999${ALEPH_SUB_PORT}
```

Library definitions
```
setenv    usm01_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm10_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm11_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm12_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm14_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm15_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm19_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm40_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm23_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm30_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm50_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm51_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm60_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
setenv    usm90_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
```

Password library
```
setenv     pw_library         USR00
```

Messaging library
```
setenv     z105_library       USR00
```

User library
```
setenv     usr_library        USR00
```

ExLibris Aleph

```
                  setenv    QUE_STARTUP_LIBS         "usm01 usm50"
```

Queues
startup
definition

```
                  setenv    WORD_STARTUP_LIBS        "usm01 usm50"
                  setenv    REQUEST_STARTUP_LIBS     "usm50"
                  setenv    ACC_AUT_STARTUP_LIBS     "usm01 usm50"
                  setenv    MESSAGE_STARTUP_LIBS     "usm01 usm50"
```

Daemons
startup
definitions

```
                  setenv    SLNP_STARTUP_LIBS        ""
                  setenv    SIP2_STARTUP_LIBS        "usm50"

                  setenv    Z39_SERVER_STARTUP    Y
                  setenv    Z39_GATE_STARTUP      Y
                  setenv    OCLC_SERVER_STARTUP   Y
                  setenv    NCIP_SERVER_STARTUP   Y
```

Servers
startup
definitions

```
                  setenv    ORACLE_SID        aleph1
                  setenv    ORACLE_OWNER      oracle
                  setenv    ORACLE_VERSION    11
                  setenv    ORACLE_CONF       ${ALEPH_MOUNT}/ora_aleph
                  setenv    NLS_LANG          American_America.UTF8
                  setenv    ORACLE_HOME       /aleph/app/oracle/product/102

                  setenv    aleph_db          ${ORA_HOST}.${ORACLE_SID}
```

Oracle
definitions

```
                  setenv  aleph_dev     ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
                  setenv  alephm_dev    ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY

                  #   setenv   alephe_dev ${USER_MOUNT}/u${ALEPH_VERSION}_$ALEPH_COPY

                  setenv  alephe_dev         /exlibris/aleph/u18_1
                  setenv  alephe_synch_dev ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY
                  setenv  aleph_utf    /tmp

                  setenv aleph_product ${ALEPH_MOUNT}/a${ALEPH_VER}_$ALEPH_COPY/product
                  setenv    TMPDIR     ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY/tmp
                  setenv    LOGDIR     ${ALEPH_MOUNT}/a${ALEPH_VERSION}_$ALEPH_COPY/log
```

ALEPH
environment

```
                  setenv    apache_dir          ${aleph_product}/local/apache
                  setenv    httpd_bin           ${aleph_product}/local/apache/bin
                  setenv    httpd_root          ${alephe_dev}/alephe/apache
```

Apache
definitions

```
                  #  Use aleph_start.private to define customer libraries

                          if (-f  $alephe_dev/alephe/aleph_start.private) then
                                  source $alephe_dev/alephe/aleph_start.private
                          endif
```

Additional
customization

Note: Only lines that are above the following line can be configured:

```
        ### End of version and/or platform dependent setting ####
```
Everything under this line is hard coded.

3.3 Library Structure

Every ALEPH library (`xxx01`, `xxx50`, `xxx60`, etc.) has a separate root directory. The term "library" in ALEPH refers to three things:

- The physical library building with books and other materials
- An Oracle database user with tables of data
- A UNIX directory structure

Here is an example of a standard UNIX directory structure. Under the library root directory there are files and directories.

The primary files are:

`file_list`   Configuration file for Oracle tables, indexes and objects. Refer to the ALEPH 500 System Administration Guide – Oracle for more details.

`prof_library`   Library logical assignments

The primary directories are:

`files/`

                        Export files (`XXX01.exportSEQ.tar.Z`)
                        Dump files (`znn.seqaa`)
                        Batch queue management
                        (alias `df1`)

`tab/`   Parameter and configuration tables (alias `dt`).

`scratch/`   Scratch directory. Used for intermediate and log files (alias `ds`).

`pc_tab/`   Parameter and configuration tables.

`print/`   Print files (alias `dp`).

4 Library Utilities

Each library's database is implemented within Oracle as a separate Oracle user. Each Oracle user owns a set of tables which contains the library's data.

Each library has access to the ALEPH utilities by activating the `util` command from the prompt. The **UTIL** main screen will appear:

```
Library Utilities
      ==================
  A. File Administration and Building
  C. Monitor Batch Jobs
  D. Online Store/Restore Administration
  E. Monitor Background Jobs
  F. View Procedures and Files
  G. Tables for Defining Database Structure
  H. Library and Installation Report (Files, Tables and Definitions)
  I. Formatting Data (PC, WWW, Reports)
  J. Web and Server Configuration
  K. ILL Tables
  L. GUI Tables
  M. GUI CATALOGING Tables
  N. Z39.50 Management
  O. ORACLE Management
  P. Unicode Tables
  Q. Data Loading, Import and Export Tables
  R. Multimedia
  S. Statistics
  W. Server Management (Monitor, Stop, Start, Log Files)
  X. Clean Up
  Y. Node Management
Please select [exit]:
```

4.1 UTIL C - Monitor batch jobs

```
C. Monitor Batch Jobs
   ------------------
   0. Exit Procedure
        1. Check Library Running Process
        2. Start Library Batch Queue
        3. Stop Library Batch Queue
        4. Check Library Lock Status
        5. Lock Library
        6. Unlock Library
        7. Show Queue of Jobs Awaiting Execution
        8. Delete Entry from Queue of Jobs Awaiting Execution
        9. Alter Run Time of Job Awaiting Execution
       10. Display Logfile of a Batch Job
       11. Display Log List of Batch Jobs

Please select [exit]:
```

ExLibris Aleph

4.2 UTIL X - Clean Up

```
X. Clean Up
       ---------------------
0.  Exit Procedure
        1.  Clean alephe_scratch
        2.  Clean data_scratch
        3.  Clean data_files
        4.  Clean TMPDIR
        5.  Clean Server Log Files
        6.  Clean apache_media
        7.  Clean utf_files Directory
        8.  Clean Virtual Library (VIR01)
        9.  Clean data_print
        10. Clean ora_media_in/out


Please select [exit]:
```

These utilities delete scratch files from various directories on the Unix server.

Note: before running the utilities you might need to close ALEPH servers, batch queues and daemons.

5   Servers, Daemons, Batch Jobs and Problem Diagnosis

There are various ALEPH servers and daemons:

- www_server
- PC server
- Z39.50 servers
- OCLC server
- Self-check server (SIP2)
- ue_daemons
- Batch jobs
- NCIP Server
- ILL Server
- SRU Server

Conventional servers' port numbers are:

- WWW server (Web)   4991
- PC server (GUI)   6991
- Z39.50 server   9991
- Z39.50 gate   7991
- Generic server (OCLC server)   5771
- Self-check server (SIP2)   5331
- NCIP   5991
- ILL Server   5551
- Apache   80
- SRU Server   5661

5.1   UTIL W - Server management (Monitor, Stop, Start, Log files)

The server management utility shows you which servers are running and has dialogues for starting and stopping the servers.

```
W. Server Management (WWW,PC,Z39)
        -------------------------------
        0. Exit Procedure
        1. Monitor Servers
        2. Stop Servers
        3. Start Servers
        4. View Log File
        5. View/Update Global System status

Please select [exit]:
```

5.2   Server Logs

The logs of the various servers are written to the `$LOGDIR` directory.

The log names are prefixed with the server type. For example `pc_ser_<port>`.

The log files contain statistics and any other input from the servers. They are useful for debugging and analyzing.

When a new server is executed the old log files are renamed with a date/time extension, for example:

```
Oct 20 12:10 www_server_4991.log.2010.1210
```

Apache server logs are written to `../alephe/apache/logs`.

5.3   Starting Your Own Server for Testing

When testing or analyzing reproducible problems, it is frequently best to start your own instance of the server rather than use an existing one. (This way the log entries for your transactions are not mixed in with those for other transactions.)

For example:

Web

The syntax for starting your own Web server is:

```
www_server <server-port> <apache-port> <num-servers> [stdout]
```

For example:

```
www_server 4123 80 1 stdout
```

`stdout` specifies that you want the server output (log) to be displayed on your display terminal.

To use your own Web server enter the URL

```
http://<URL>:<server-port>/<type>
```

In our example, to use the Web OPAC enter

```
http://<URL>:4123/F
```

GUI
The syntax for starting your own GUI server is:

```
pc_server <port number> stdout
```

For example:

```
        pc_server 6123 stdout
```

`stdout` specifies that you want the server output (log) to be displayed on your display terminal.

Specify **6123** as the address in your PC's `..\Alephcom\Tab\library.ini` file, instead of `6991`.

5.4   PC Client / PC Server

Configuration of the `pc_server` is performed using **UTIL J/5 Definition of Defaults for PC Server Defaults (pc_server_defaults)**.

When a problem occurs in the GUI it can be with either the client or the server (the PC server).

Error messages may have been generated by the PC client or they may have been sent from the server. Certain kinds of messages clearly indicate a problem on the PC side. These include: `GDI failure`; `Dr. Watson's`; and `General Protection Faults (GPF)`. With such problems, make sure that the GUI client is at the same service pack level as the server you are trying to connect to and make sure that the PC has sufficient free memory and disk space.

Other error messages such as `Remote file error (global-xx)`; `Remote service error`; or `Failed to read reply` are passed from the server and indicate a problem on the server.

Here is how the PC and the PC-server interact:

5.5   Connecting

When you start the first GUI module on your PC, if you have not saved your logon identification on the client, the software prompts you for a user name and password. It checks all the servers listed in the `..\Alephcom\Tab\library.ini` file. If this user name/password isn't valid on any of the servers it can connect to, the message `Password not verified on connectable hosts` is displayed. This indicates either that this user name/password is not valid; OR that the address specified in the `..\Alephcom\Tab\library.ini` is wrong; OR that the pc_server specified in the `library.ini` entry is down.

Assuming that the password is verified, you then connect to a library. Select **File / Connect** from the main menu. The system displays the libraries listed in the module's `per_lib.ini` file, for example, `..\Circ\Tab\per_lib.ini`.
When you click on a particular library (or base), the client software goes to the `..\Alephcom\Tab\library.ini` file and tries to connect to the IP address specified for this library.

Service Requests
(**Note:** See the *ALEPH Staff User Guide*, available from the Ex Libris Documentation Center, for documentation of each GUI module.)

The functions you perform on the PC generate service requests for the PC server. The `pc_com/pc_server` program processes all incoming requests. It checks the license date, user limit, and so on, and passes control to the program specified in the service request. For example, if the service request is **c0511** and the module is **ACQ**, it passes control to the `pc_acq/pc_acq_c0511` program. Just before doing this it writes an entry to the pc_server log:

```
SERVICE:     C0511
MODULE:      Acquisition Services
DESCRIPTION: Invoice Header Information
ACTION:      GET
PROGRAM:     pc_acq_c0511
```

The text for the description comes from the `../alephm/proc/pc_service.dat` file. This file shows all available services. Each service is self-contained. Though the program may call other non-service programs in the course of its processing, it is the PC which initiates each service call.

The `pc_xxx_cnnnn` programs call the `com/service_error_message` routine, passing a specific error number as a parameter, in order to generate text for error messages. The `service_error_message` reads the `../alephe/error_<lng>` file to get the text for this program for the specific error number.

For example, if there was an error `0101` in `pc_acq_c0511`, the `service_error_message` would read the `../alephe/error_<lng>/ pc_acq_c0511` file and find this entry:

```
0101 0005 L Invoice Net Amount must be numeric.
```

(An error message such as `Error 21 Not defined for service C0204 in pc_cat_c0204` indicates that there is no line with text for the `0021` error in the `$aleph_root/error_lng/pc_cat_c0204` file when there should be.)

5.6 Analyzing PC Server Problems

There is normally just a single PC server (6991). Multiple logs for this server (for example, `pc_ser_6991`) in a time period when the server was not stopped intentionally (for example, for a backup) indicate a server failure. To get more information than what you see in the `pc_ser_nnnn` file type:

```
pc_server view <port number> <number of lines>
```

For example

```
>>pc_server view 6991 25
```

(The fact that the PC server is stateless means that each transaction is self-contained. The transaction can be caught and simulated without a client.) When you get the number of the problem `IN` transaction, such as `55459 IN`, type:

```
pc_server check[x] <port number> [<line number>]
```

For example

```
pc_server check 6991 55459
```

or (if the DATA line you see is truncated and you want to see the rest):

```
pc_server check[x] <port number> [<line number>]
```

For example

```
>>pc_server checkx 6991 55459
```

Note that in cases where the transaction is performing an update, execution of `pc_server check` will cause the update to be performed again.

Example:

The system does not have a catalog package for some reason, that is, `$data_root/pc_tab/catalog/pc_cat.pkc` is missing. When connecting to the **Cataloging** module the error message `Failed to read reply` appears. This means that the server could not handle the abnormal situation. Type:

```
M201>> pc_server view 6991 10
```

and you will see:

```
1132   IN   2710   Mon   21-11-2011   14:44:33   010.001.049.132
010001049132TEST;TEST;default;25bdd562-2b3e-4788-b282-
7d57d0d1b9fc;CAT;N; C0201 NEW MAB01GERCITEST 000000
```

This reveals that transaction number `4078` called function C0513 (Acquisition Services).

For more details use:

```
M201>> pc_server view 6991 10
```

and as part of the output you will see:

```
SERVICE     : C0201
MODULE      : Catalog Services
DESCRIPTION: Connect and Check Tables Date
ACTION      : NEW
PROGRAM     : pc_cat_c0201

Load: /exlibris/aleph/a20_3/alephe/tab/tab_system_status

check_permission debug mode is on

check_permission CITEST     MAB01(     )                      >
- v
```

```
I/O                error                :               file
'/exlibris/aleph/a20_3/mab01/pc_tab/catalog/pc_cat.pck'
error code: 3/5 (ANS85), pc=0, call=1, seg=0
 13    File not found
```

Now it is obvious that **UTIL M/7 Update Tables Package** should be activated in order to create the catalog package.

Note: Even if you cannot understand the error, it is best to catch the transaction and put it aside for further handling by Ex Libris Support. This can be done with the following sequence of commands:

```
ALEPH>> cd $LOGDIR
ALEPH>> tail -10 pc_ser_6991 > my_transaction_file
```

Note:
To use the `pc_servet view` and `pc_server check`, set the "`pc_transactions_log`" variable to "Y" in "`pc_server_defaults`".

5.7   PC Server Configuration (pc_server_defaults)

5.7.1   Max Response Time

If you have a problem with transactions timing out, increasing the `alephe/pc_server_defaults PC_SERVER_MAX_RESPONSE_TIME` value might help.

The only downside is that problematic transactions (transactions which are looping, etc.) may run even longer. So before changing the `PC_SERVER_MAX_RESPONSE_TIME` value, make certain that the problematic transactions are not due to a lack of the appropriate Oracle indexes: compare your `xxxnn` library's `file_list` to the `usmnn`'s `file_list`.

5.7.2   Backend Servers

The `alephe/pc_server_defaults  PC_NUM_SERVER` parameter determines how many backend servers are running. The default value is 5.

5.8   ue daemons

Certain utilities can be initiated to execute as daemons, constantly checking to see if work needs to be done. These are:

- **ue_01 indexing daemon**
- **ue_06 request handling**
- **ue_08 update BIB ACC from AUT**
- **ue_11 messaging daemon**
- **ue_21 Publishing Utility**
- **ue_13 OAI Harvester**
- **ue_17 ILL Server Daemon**
- **ue_19 ILL Automatic Processing**

5.9   Batch jobs

(**Note:** For detailed information on batch jobs which build indexes, please consult the document, *How To Run Index Jobs* available on the [Ex Libris Documentation Center](#).)

Batch jobs are executed either periodically (daily, weekly, etc.) or on request. They may be initiated via the **Services** menu in the GUI or the utilities **UTIL E/15 Managing Job Daemon** and **UTIL E/16 Update Daemon Job List**.

5.10 Services

Each GUI module has a **Services** menu. This option lets you submit batch jobs from a menu, choosing the values you want for each parameter. The submission screens which you see are in the `../alephe/pc_b_<lng>` directory (unless directed elsewhere in `path_convert`) and may be modified. As is the case with the `www_a_<lng>` Web OPAC screens, the system checks first for a suffixed version and then looks for a non-suffixed version.

When you submit the job from a particular window, the system calls the corresponding program to process the service. If errors are found in the parameters, an error message will be displayed. Otherwise the message `Job submitted to queue` will be displayed.

The job will then be placed in the batch queue of the library for which it was submitted. If the library's batch queue is running, then the job will be executed and will show up under **UTIL C/1 Check Library Running Process** as being executed.

The job will appear in **UTIL C/1 Check Library Running Process** with its parameters. In this example the batch service is **cir-51 Overdue and Lost Billing Summary Letter**:

```
aleph  31696  59116    43  18:24:25  pts/12    0:00  csh  -f
/exlibris/aleph/a18_1/aleph/proc/p_cir_51
EXU50,ovrdustaf,N,00,00,00,00,00,00,00,00,00,00,N,00,00,00,00,0
0,,00,14,00,Y,Y,N,O,4,0,00,
```

The `p_cir_51` component is the procedure which is being executed. The procedures are in the `aleph/proc` directory. `../aleph/proc/p_cir_51` executes three different programs:

```
cobrun b_cir_51_a
cobrun b_cir_51_b
cobrun b_cir_51_e
```

Print templates are stored in `usm01/form_<lng>`. All formats are taken from XSL files.

Any printed output produced by the batch job will be in the library's print directory (for example, `../usm50/print`). The name of the output file is controlled by the **Output File** value on the submission screen. The log of the job will appear in the

`../alephe/scratch` directory under the name of the process (in our example, `usm50_p_cir_51.00138`).

For more information on forms, see the document *Customizing Printouts (XML and XSL) - 17.pdf* and the *Printing* chapter in the *ALEPH Staff User's Guide* available on the Ex Libris Documentation Center.

For detailed information on batch jobs which build indexes, see the document, *How To Run Index Jobs* available on the Ex Libris Documentation Center.

5.11 Job List

Jobs which need to be run on a periodical basis can be placed in the `job_list` file. Use **UTIL E/15 Managing Job Daemon** and **UTIL E/16 Update Daemon Job List**.

5.12 cron Jobs

"cron jobs" are similar in principle to the ALEPH `job_list` described in the preceding section, except that their control and maintenance are outside of ALEPH.

5.13 www_server (Public)

The WWW server for Web OPAC is accessed via `http://<URL>/F`. For example: `http://ram11:8991/F`.
The Web OPAC is an interface for accessing and searching an ALEPH 500 online catalog via the HTTP Internet standard. The Web OPAC allows a patron either to enter the system as a guest user, or to sign in, thereby activating his customized profile.

UTIL J - Web and Server Configuration is used for several `www_server` configurations. This includes messages to Web OPAC users (`www_f_heading`); sort types for display in Web OPAC (`www_f_sort_heading`); and definitions for server defaults (`www_server.conf`).

The Web OPAC screens are in the `./alephe/www_f_eng` directory. See *List of Web OPAC HTML Files* for a description of each file and *Web OPAC Screens – Structure and Customization* explains how to customize the screens. These documents are available on the Ex Libris Documentation Center.

5.14 www_server (Staff)

The WWW server **Staff Menu** is accessed via the address `http://<URL>/S`. For example, `http://ram11:8991/S`:

OPAC – Access to the ALEPH 500 online catalog.

Ex Libris Documentation Center - a link to the Ex Libris Documentation Center.

Utilities – Utilities that are used to view/update the Chinese dictionary (the dictionary is used for translation for filing and for word breaking) and to view/update Synonym groups (Synonym groups are optional for word indexing).

5.15 Z39.50 Servers

UTIL N - Z39.50 Management is used for Z39.50 gate and server configuration. See the Z39.50 documentation available on the Ex Libris Documentation Center.

5.16 OCLC Servers

OCLC records can be dynamically loaded into ALEPH from a PC connected to OCLC with the **OCLC Passport**, **OCLC CatME** or **OCLC Connexion** software. Please consult the document, *How To Load OCLC Records into ALEPH* available on the Ex Libris Documentation Center.

5.17 SIP2 Server (3M Standard Interchange Protocol)

This protocol provides a standard interface between a library's Automated Circulation System (ACS) and library automation devices.

Self-check systems enable library patrons to borrow and return materials without the assistance of library staff, usually by means of scanning the barcodes on the item and on the patron's library card.

See the document, *How To Set Up a SIP2 Server for Self-Check Systems* available on the Ex Libris Documentation Center.

5.18 NCIP Server (NISO Circulation Interchange Protocol)

This standard defines a protocol that is limited to the exchange of messages between and among computer-based applications to enable them to:

- Perform the functions necessary to lend and borrow items.
- Provide controlled access to electronic resources.
- Facilitate co-operative management of these functions.

See the document, *NISO Circulation Interchange Protocol Server NCIP Server* available at http://www.niso.org/schemas/ncip/.

5.19 Interlibrary Loan Server (ILL Server)

The Interlibrary Loan Server is a server that participates in handling ISO ILL transactions. It is essential for handling ILL processes that support ISO 10160 and 10161 standards via the ILL module.

The ILL Server performs three major tasks in the process of handling an ISO ILL transaction:

1. Receiving the ILL APDUs (Application Protocol Data Units) and decoding them.
2. Storing the packets received by the local server, as indexed documents in the `ILLSV` library
3. Logging APDUs sent from the local server. This is also done in the `ILLSV` library.

The ILL Server is the link between the ILL module and the communications system by which the ILL APDUs are delivered, translating protocol packets into library documents.

5.19.1 Running the ILL Server

The ILL Server is managed in the same manner as other servers, such as the PC Server and the WWW Server, using **UTIL W Server Management**. In each procedure, the ILL Server is listed under **Other Server**.

5.19.2 The ILLSV Library

Once ISO ILL APDUs have been received and decoded they are written as documents into a library named `ILLSV`. This library stores the packets received by the local server as indexed documents.

There are two types of fields in the record:

1. Fields with a numeric tag – These are the bibliographic fields expanded from the APDU.

    Example: `245   L $$aSpiders of Britain`

2. Fields with the **ILL** tag. These fields are the exact fields of the APDU. They always have a subfield **a** with the name of the tag in the APDU, and a subfield **b** with the value.

    Example: `ILL   L $$aIllRequest.ClientId.ClientName$$bABEP`

Records logged in the ILL Server that have a **466** field with a subfield **a** value of LOG, are APDUs sent from the local server. For example:
`466 L $$aLOG$$bREQUEST$$c20031026`
All other records are APDUs received from other servers.

5.20 SRU Server

SRU (Search/Retrieve via URL http://www.loc.gov/standards/sru/) is a standard search protocol for Internet search queries, utilizing CQL (Common Query Language), a standard query syntax for representing queries.

An SRU request is a HTTP URL. It consists of a base URL and a search part, separated by a question mark. The search part consists of parameters separated by an ampersand, each with structure "key=value".

For example, consider the SRU request:

http://localhost:5661/usm01?version=1.1&operation=searchRetrieve&query=dinosaur

The base URL is:

http://localhost:5661/usm01 and the search part is:

`version=1.1& operation=searchRetrieve&query=dinosaur`.

The response to an SRU request is an XML document.

5.21 Apache

5.21.1 Configuration

Apache configuration files are placed in the `$alephe_root/apache/conf` directory. The main apache configuration file is:
```
-rw-r--r-- 1 aleph exlibris  9454 May 31 10:46 httpd.conf
```

The Apache SSL configuration file is:
```
-rw-r--r-- 1 aleph exlibris  5177 May  8 10:57 ssl.conf
```

5.21.2 Start/Stop

Use the scripts below to start and stop Apache.

5.21.3 Startup:
```
cd $alephe_root/apache/bin
./apachectl start
```

5.21.4 Shutdown:
```
cd $alephe_root/apache/bin
./apachectl stop
```

Note:
Customers running Apache as root must use the `. /apachectl_auto` command, placed in the same `$alephe_root/apache/bin` directory, instead of `apachectl start`.

5.21.5 Monitoring

Use the Unix `ps` command with the following flags:
* For Linux:
  ```
  ps -efw | grep http | grep aleph
  ```
* For SUN or AIX:
  ```
  ps -ef | grep http | grep aleph
  ```

The output of this command displays several http processes:

```
aleph    18401 11964  0 Jun08          00:00:00
/exlibris/aleph/a21_1/product/local/apache/bin/httpd -d
/exlibris/aleph/u21_1/alephe/apache
aleph    19052 11964  0 Jun08          00:00:00
/exlibris/aleph/a21_1/product/local/apache/bin/httpd -d
/exlibris/aleph/u21_1/alephe/apache
```

5.21.6 Log files

Apache log files are placed in the `$alephe_root/apache/logs` directory.
The Apache access log file can be updated by any connection to apache:
```
$alephe_root/apache/logs /access_log
```

Apache SSL access log:
```
$alephe_root/apache/logs /access_log_ssl
```

The Apache error log file is updated by apache start or stop or if any error occurs while apache is running:
```
$alephe_root/apache/logs /error_log
```

Apache SSL error log:
```
$alephe_root/apache/logs /error_log_ssl
```

The Apache PID file contains a process ID for the currently running apache. This file exists only when apache is up and running; it is removed by apache shutdown and created again by apache startup:
```
$alephe_root/apache/logs / httpd.pid
```

6 Automatic Startup/Shutdown of All Aleph Components

The mechanism for the automatic start up and shut down of all Aleph components is implemented with the initial Aleph installation.

The automatic startup/shutdown mechanism is activated by the `/etc/init.d/exlibris` script with links created in special OS directories.

- For Red Hat Linux OS:
```
lrwxrwxrwx 1 root root 18 Jan  4  2010 /etc/rc0.d/K10exlibris ->
../init.d/exlibris*
lrwxrwxrwx 1 root root 18 Jan  4  2010 /etc/rc1.d/K10exlibris ->
../init.d/exlibris*
lrwxrwxrwx 1 root root 18 Jan  4  2010 /etc/rc2.d/S98exlibris ->
../init.d/exlibris*
lrwxrwxrwx 1 root root 18 Jan  4  2010 /etc/rc3.d/S98exlibris ->
../init.d/exlibris*
lrwxrwxrwx 1 root root 18 Jan  4  2010 /etc/rc4.d/S98exlibris ->
../init.d/exlibris*
lrwxrwxrwx 1 root root 18 Jan  4  2010 /etc/rc5.d/S98exlibris ->
../init.d/exlibris*
lrwxrwxrwx 1 root root 18 Jan  4  2010 /etc/rc6.d/K10exlibris ->
../init.d/exlibris*
```

- For SUN Solaris OS:
```
lrwxrwxrwx  1 root  root  20 Sep  8  2010 /etc/rc0.d/K10exlibris
-> /etc/init.d/exlibris*
lrwxrwxrwx  1 root  root  20 Sep  8  2010 /etc/rc2.d/S96exlibris
-> /etc/init.d/exlibris*
```

Note:
All files and links in the server root directories are created automatically during the initial Aleph installation and should not be changed or removed.

The tool performs a startup/shutdown of each Aleph component placed in the `/exlibris/startup` directory and contains the following components:
- `init.dat` – the configuration file
- `start_stop` – the startup/shutdown tool

6.1 Startup Configuration

Perform any changes necessary for maintenance of startup and shutdown Aleph components in the `/exlibris/startup/init.dat` configuration file only. We highly recommend that you save a side backup of this file before making any changes.

The following is the structure of the `init.dat` file:
```
# oracle
Y:oracle:oracle:/exlibris/app/oracle/product/1122
```

```
# apache
Y:apache:aleph:httpd:/exlibris/aleph/u21_1:../a21_1:apachectl_auto
:apachectl

# aleph
Y:aleph:aleph:/exlibris/aleph/u21_1:aleph_startup:aleph_shutdown
```

This file contains setup definitions for the startup/shutdown of each component (Oracle DB and listener, apache, and Aleph application services) in the relevant section. The setup includes the following fields:
- Activate or not startup/shutdown mechanism of the component
- Component name
- Owner (Unix user name)
- Placement
- Startup routine name (for apache and application only)
- Shutdown routine name (for apache and application only)

In the example above, the setup values for Aleph are the following:

Activate or not startup/shutdown mechanism for the component	`Y`
Component name	`aleph`
Owner (Unix user name)	`aleph`
Configuration placement	`exlibris/aleph/u21_1`
Startup routine name (for apache and application only)	`aleph_startup`
Shutdown routine name (for apache and application only)	`aleph_shutdown`

To exclude one or more component (Oracle DB and listener, apache, and Aleph application services) from the automatic startup/shutdown mechanism, edit the `/exlibris/startup/init.dat` file.

The following is an additional example of two Aleph copies installed on the same server, when only copy 1 should activate automatically:

```
# oracle
Y:oracle:oracle:/exlibris/app/oracle/product/1122

# apache
Y:apache:aleph:httpd:/exlibris/aleph/u21_1:../a21_1:apachectl_auto
:apachectl
N:apache:aleph:httpd:/exlibris/aleph/u21_2:../a21_2:apachectl_auto
:apachectl

# aleph
Y:aleph:aleph:/exlibris/aleph/u21_1:aleph_startup:aleph_shutdown
N:aleph:aleph:/exlibris/aleph/u21_2:aleph_startup:aleph_shutdown
```

6.2   Manual Startup/Shutdown

Manual startup/shutdown for all components can be done using the mechanism described above.
Connect as root user:

```
su - root
```

To activate the shutdown command:
```
cd /exlibris/startup
./start_stop stop
```
Wait until all components are shut down and perform relevant maintenance.

To activate the startup command:
```
cd /exlibris/startup
./start_stop start
```

Wait until all components start up and check the application.

To start up or shutting down each component separately, refer to the following documents:

- Oracle – Refer to the **UTIL O/1/1** and **UTIL O 2/2** sections of the *Aleph System Administration Guide*.
- Apache – Refer to the Apache section of the *System Administrator's Guide - System Overview*.
- Aleph processes – Refer to the **Library Utilities** and **Servers, Daemons, Batch Jobs and Problem Diagnosis** section of the *System Administrator's Guide - System Overview*.

The following is the correct order for starting up all components:
1. Oracle
2. Apache
3. Application

Be sure that each component is up and running before proceeding with the next one.

The following is the correct order for shutting down of all components:
1. Application
2. Apache
3. Oracle

Be sure that each component is down before proceeding with the next one.