ASTRA/LISTSERV DATABASE SYSTEM LSVDBEAR DRIVER: INSTALLATION AND REFERENCE

CNUCE 227 - 91

June 1991

Esra Delen
Silvia Giordano
Giuseppe Alberto Romano
Daniele Vannozzi

ASTRA Working Group

ASTRA@ICNUCEVM.CNUCE.CNR.IT

ASTRA/LISTSERV Database System
LSVDBEAR Driver: Installation and Reference

CNUCE ZC 227/91

Esra Delen - esra@icnucevm.cnuce.cnr.it
Silvia Giordano - silviag@icnucevm.cnuce.cnr.it
Giuseppe A. Romano - romano@icnucevm.cnuce.cnr.it
Daniele Vannozzi - vannozzi@icnucevm.cnuce.cnr.it

CNUCE - Istituto del CNR.
Via S. Maria 56124 Pisa - Italy.

Abstract

EARN documents are the most important documents of the EARN network. These documents, in brief, are the minutes of the meetings, the proposals or the official papers. Until now these documents were managed on a LISTSERV. The access to these databases were only by sending commands to the related LISTSERV. Now the ASTRA Group has developed a common interface and a driver to access these documents via ASTRA.

In this booklet, we describe the driver to access the EARN documents via ASTRA. Both the user guide to the driver and the technical description and specification of the driver are included in the booklet. For more detailed information or for further questions, please consult the ASTRA Group (ASTRA@ICNUCEVM).

Table of Contents

ASTRA-LISTSERV Databases Description 1
ASTRA-LISTSERV Database Installation Guide 4
ASTRA-LISTSERV Databases Access 10
ASTRA Proposal for EARN Documentation 13
LSVDBEAR LISTSERV driver: functional specifications. 16
ASTRA-LISTSERV driver: how to change the keywords. 21
ASTRA-LISTSERV database: an example. 22
- The LSVDBEAR dbtable at CNUCE 22
- The ASTRA public documentation ACCESS file 22
- The ASTRA public documentation LIST file 22
- The ASTRA public documentation FILELIST file 22

ASTRA-LISTSERV Databases Description

The LISTSERV driver named LSVDBEAR has been written to use the LISTSERV database functions for different types of documentation. Infact this function allows a list of NOTEBOOK files or the BITEARN, PEERS or LISTS files to be used as a database.
The new driver allows every type of lists of files to be used as a database.
This driver uses the DBNAMES and DBINDEX files in which it records the main keywords and their values.
These keywords have been chosen according to the EARN document description - Guidelines for the Production of EARN Documents by P. Bryant and H. Nussbacher (EXEC106 89).
This driver can use a different access mode for every file.

```
Example:
  The list  TEST  contains two different types of
  files,   one  accessible  by   everyone and one
  accessible only  by  a restricted group of users.
  In the file TEST ACCESS, used   by  the driver,
  these files must be differentiated:
  The files accessible  by   everyone are defined
  public (PUB) and the private  files are defined
  private (PRV).
```

A query to the databases managed by this driver can be sent using the LISTSERV facility (LDBASE) or sending a request mail to the Listserv machine, or directly by using the ASTRA User Interface.
The ASTRA Interface allows the users to query the databases using the LISTSERV database language or the ASTRA User Interface language. It is also possible to select directly the LDBASE tool.
To make these capabilities available for an ASTRA user, an interface between the LISTSERV language and the ASTRA language has been introduced.

The interface translates a Listserv query in an ASTRA query in some steps:

- Identifies the parts of the query
- Resolves the text operators
- Resolves the binary operators
- Recomposes the whole query

```
If the query is:
   Query #1: Sommani.author. and (ASTRA.title xor
             TCP-IP)

The interface identifies two parts:

   Part1: Sommani.author. and (ASTRA.title xor TCP-IP)
   Part2: (ASTRA.title xor TCP-IP)

There are two text operators:

   Arg1: Sommani.author.
   Arg2: ASTRA.title
They are solved in:
   Arg1: where author contains Sommani
   Arg2: where title contains ASTRA

The query now is:
   Query1: where author contains Sommani and (where
           title contains ASTRA xor TCP-IP)

There is only one meaning operator:
   OP1: and
Infact "xor" cannot be solved because its
arguments are not comparable in the Listserv
Language and it is omitted.

The query becomes:
   Query1: Search TCP-IP in EXEC where author
           contains Sommani and title contains ASTRA
```

Some problems have been solved because of the differences between the two languages. Infact the ASTRA language is more powerful and some operators have been translated in one less restricted; on the other hand the LISTSERV language allows the phonetic query qualification, which cannot be made with the ASTRA language.

The search rules of the LISTSERV language are substring-based rules, when a user makes a query, the result is all the documents which contain at least a word that contains the substring requested.
If a user searches BALL, he will receive every document containing: FOOTBALL, BASEBALL, BALLAD, BALLET, etc... The LISTSERV language permits also to have a more exact research using the single or the double quotes.
If a user searches ″BALL″, he will receive every document containing BALL and not ball or FOOTBALL.

The Interface has been made to consider two kinds of research:

- A normal research
- An advanced research

A normal research is made without thinking about the case of the words or if a word can be plural (search ball to have: Ball, balls, etc..).
An advanced research, on contrary, is made thinking about the case of the words or if a word can be plural (search ″ball″ to have only ball).
The advanced research is obtained througt the text operators (.F/C. , .L/C. , etc..).

```
If the query is:
 Query #1: sommani.F/C. and TCP-IP.title.
The query becomes:
 Query1: Search "Sommani" in EXEC where title contains TCP-IP

While, if the query is:
 Query #1: sommani and TCP-IP.title.
The query becomes:
 Query1: Search sommani in EXEC where title contains TCP-IP
```

The following table shows the logic of translation used by the ASTRA-Listserv Interface.

AQUARIUS Language Search Command	Listserv Language Search Command Translation
a	a
a ADJ b	a b
a ADJn b	a b
w ADJ x y z	w x y z
a AND b	a AND b
a b	a OR b
a NEAR b	a AND b
a NEARn b	a AND b
a NOT b	a AND NOT b
a NOTSAME b	a AND b
a NOTWITH b	a AND b
a OR b	a OR b
a SAME b	a AND b
a SYN b	a AND b
a WITH b	a AND b
a XOR b	a AND NOT b OR b AND NOT a
a$	a
a$n	a
aa.F/C.	"Aa"
aa.L/C.	"aa"
aA.M/C.	"aA"
a.par.	where par contains a
aa.U/C.	"AA"
aa..F/C.	"aa"
aa..L/C.	"AA"
aA..M/C.	"aA"
a..par.	where par not contains a
aa..U/C.	"aa"
MASK a	a
MASK a n	a
MASK a TO b	a AND b
ROOT operand	a

ASTRA-LISTSERV Database Installation Guide

Introduction

With the LISTSERV database feature, it is possible to make searches and to construct queries on the NOTEBOOK, BITEARN, LISTS and PEERS files.

Until now the EARN documents i.e., the minutes of meetings, the proposals, and the papers of Executive Committee and Board of Directors were available only by a file server, namely a certain LISTSERV/LISTEARN at a central node. Thus it was hard to make a search according to keywords or specific paragraphs of the documents.

The ASTRA - LISTSERV interface is developed to overcome this situation. By the help of the new driver installed on a LISTSERV/LISTEARN, it is possible to search EARN documents by keywords with the ASTRA user interface and the ASTRA query language.

This documentation describes the database's installation steps. It contains all the information to install a new database of EARN papers (EXEC + BOD). If a common header desciption is adopted for any kind of EARN documents (papers, manuals, etc) this installation procedure will be generailized.

LISTSERV Database

A LISTSERV database is a set of files to be defined and a driver is a program accessing those files. Generally users need not to be aware of where the files are or what the driver is. There is a common interface for the users to access to all databases transparently.

EARN databases on LISTSERV are the collection of EARN Executive Committee and EARN Board of Directors documents. These documents normally are accessible through the file server feature of LISTSERV. Now it is also possible to access them by the Database function of LISTSERV. To make this possible first a database containing EARN documents must be installed on a LISTSERV.

Installation guide

To install a database on LISTSERV, there have to be three files available. These files are:

```
<DBname>  LIST       A
<DBname>  FILELIST <diskmode>
<DBname>  DBNAMES  <diskmode>
<DBname>  DBINDEX  <diskmode>
<DBname>  ACCESS     A
```

The basic steps to install a database are:

Step 1

Create the file < DBname > LIST A. This file is necessary for controlling the access level to the database. The header of such a list can be:

```
*  <Description of the list>
*
*  REVIEW= PUBLIC         SUBSCRIPTION= CLOSED      SEND= PRIVATE
*  NOTIFY= YES            REPLY-TO= LIST,RESPECT    FILES= YES
*  VALIDATE= ALL          STATS=EXTENDED,OWNER      ACK= NO
*  CONFIDENTIAL= NO       LOOPCHECK= NOTCOUNT       RENEWAL= NONE
*  NOTEBOOK= NO,<diskmode>,SINGLE,PRIVATE           ERRORS-TO= OWNER
*  FORMCHECK= NO
*  OWNER= <owner net address>
```

Here the necessary keywords are:

- SUBSCRIPTION= CLOSED; this is to avoid people subscribing to the list because this is not a REAL list
- SEND= PRIVATE; this is to avoid people sending mail or files to the list because this is not a REAL list
- VALIDATE= ALL COMMANDS; this is only for the owner to send commands to the list, protected by the password
- NOTEBOOK= NO,<diskmode>,SINGLE,PRIVATE; here the <diskmode> is the same diskmode of <DBname> DBNAMES and <DBname> DBINDEX files; the notebook is SINGLE because only one archive for the documents; it is PRIVATE to make the database accessible only by the OWNER and the subscribers of the list

More information about the list header keywords can be obtained from LISTSERV/LISTEARN with the command GET LISTKEYW MEMO.

Step 2

Create the file <DBname> FILELIST * on LISTSERV. With this filelist the documents in the database, the people who can access the database and their access levels can be defined. An example of such a filelist can be:

```
*  <name and brief description of the filelist>
* :::::::::::::::::::::::::::::::::::::::::::::::::::::::::
*
*  The GET/PUT authorization codes shown with each file entry
*  describe who is authorized to GET or PUT the file:
*
*     ALL = Everybody
*     N/A = Not Applicable
*     LCL = Local users, as defined at installation time
*     PRV = Private, ie list members
*     OWN = List owners
*     NAD = Node Administrators, ie official BITNET/EARN contacts
*     CTL = LISTEARN Controllers (Also called "Postmasters")
*
*+ ZAP= <userid@nodeid>
*+ BEN= <userid@nodeid>
*
* :::::::::::::::::::::::::::::::::::::::::::::::::::::::::

*************************************************************
*                                                           *
* FILELIST declarations                                     *
*                                                           *
* Those filelists  which have a PUT code of N/A are  either *
* maintained automatically by  LISTSERV or come  in as part *
* the LISTSERV software package and  can  therefore not  be *
* changed by the installation.                              *
*                                                           *
*************************************************************
*                            rec                 last - change
* fn ft    GET PUT -fm lrecl nrecs   date    time   File desc.
* -- --    --- --- --- ----- ----- ------- -------- ----------

 <fn ft>   PRV ZAP V      73     75 90/05/08 13:12:27
 <fn ft>   BEN ZAP V      73     75 90/05/08 13:12:27
```

The guidelines and procedures of how to create and define a filelist can be obtained from any LISTSERV/LISTEARN server with the command:

```
GET LISTFILE MEMO
```

Step 3

Create two dummy files:

```
<DBname>  DBNAMES <diskmode>
<DBname>  DBINDEX <diskmode>
```

on the same disk specified by the <diskmode> parameter of the NOTEBOOK keyword on <DBname> LIST A header. This disk must be R/W accessible by the database driver.

```
<DBname> DBNAMES <diskmode>  file:
----------------------------------
```

This file is first created as a dummy file. The correct DBNAMES file is created automatically by the driver the first time the driver is run. Then if there is an update to the database the files are updated automatically by the driver. The format of this file must be fixed (F) and the lenght of the records must be 80 characters.

An example <DBname> DBNAMES <diskmode> file is:

```
1 EXEC119 89 J5"109
2 EXEC106 89 J5"165
3 EXEC156 89 J5"605
4 EXEC26 90 J5"75
5 EXEC58 90 J5"92
```

Each line in this file represents one document of the database. Here the first column represents the symbolic name of the file. Then there are the file name, file type and the file mode of the database files.

' " ' is a separator character used to separate the file id and the length of the file.

```
<DBname> DBINDEX <diskmode> file
--------------------------------
```

This file is first created as a dummy file. The correct DBINDEX file is created automatically by the driver the first time the driver is run. Then if there is an update to the database the files are updated automatically by the driver.

An example <DBname> DBINDEX <diskmode> file is:

```
0 890821 FA ST"FT EARN"FN EXEC119 89"FS D"FM ST"FX UR"PB 13"PH 1 11
```

The fields of the <Dbname> DBINDEX <diskmode> file are derived from the EXEC standard format. In this file each line denotes one file.

The fields of each line are:

1. The symbolic name of the file. This file may be referred to as the document.

2. The starting line of the document in the file indicated on the first field.

3. The ending line of the document in the file indicated on the first field.

4. The date of the document. The date specified in the header of the document.

5. The author tag (FA) and the author name (ST in our example)

6. The file title tag (FT) and the file title (EARN in our example)

7. The file name tag (FN) and the file name of the document (EXEC119 89 in our example) This file name is the name of the file as it appears on LISTSERV disk.

8. The status tag (FS) and the status of the document (D in our example)

9. The file maintainer tag (FM) and the name of the file maintainer (ST in our example)

10. The access tag (FX) and the access level of the document (UR in our example)

11. The start of the body

12. The start and finish of the header

Step 4

Create the < DBname > ACCESS A file. This file contains the ordered list of the files contained in the database and the access level for each file. This is an example:

```
EXEC1     85          ALL
EXEC1     87          EXE
EXEC1     88          ALL
EXEC1     90          ALL
EXEC1     91          EXE
EXEC10    88          EXE
```

The fields of each line are:

1. The filename

2. The filetype

3. The access level (See later: Step 6.6)

< DBname > ACCESS is the default name. If it is preferred to use another name, it must be defined in the LSVDBEAR DBTABLE (See later: Step 6).

Step 5

Add a record to the DATABASE FILE A on LISTSERV. An example of such a line could be:

```
BITEARN LSVDBBN
LISTS LSVDBLS
PEERS LSVDBPN
EXEC  LSVDBEAR
BOD  LSVDBEAR
```

The line shows the association between the database name and the driver module.

Step 6

Add a line to the file LSVDBEAR DBTABLE *. This file shows the mask describing the kind of the fileid to be accessed by the driver, for every database. An example of this file is:

```
EXEC EXEC####   ## - 50 PUB
BOD BOD#####  ## - 50 ALL
PIPPO * NOTE????  - 50 PRV PIPPO PAPPO
```

The fields specified in this file are:

1. The name of the database.

2. The filename mask to be accessed in the database denoted in the first field.

3. The filetype mask to be accessed in the database denoted in the first field.

4. The character used to compose the banner that separates the header and the body.

5. The length of the banner.

6. The access level of the database files. This access level can be

```
PUB = Public
ALL = Everybody
PRV = Private, ie list members
<fac>= Predefined FAC
```

If the access level is omitted, the default value is PUB. The rules on how to define a FAC can be obtained from the nearest LISTSERV, by the command:

```
GET LISTFILE MEMO
```

7. This field denotes the filename of the file to be used to control the access level. The file type of this file can be specified in the eighth field. The last two fields are optional. If omitted, the default values are < DBname > for the filename and 'ACCESS' for the filetype. This file must be sorted in ascending order, according to the filename and filetype.

Privacy

The privacy is controlled by the driver with the help of the access control file. The driver controls the access level of single file from the access table and the access level of the database from the filelist (where the FACs are defined). If two access levels match, then the file is accessible by the user.

If FAC = PUB in the < DBtablename > DBTABLE * file then no control is done for the privacy. If the FAC = ALL in the mentioned file, then all files having FAC = ALL in the < DBname > ACCESS * file are accessed. If the FAC = PRV in the < DBtablename > DBTABLE * file, then all the files with FAC different than ALL in the access file are accessed by the driver. For all other defined FACs the file is accessed if the FAC of this file in the ACCESS file is the same of the FAC specified in the DBTABLE file.

Access levels

For having more levels of access in a collection of documents (EXEC or BOD), more databases must be defined. For the private documents a private database can be defined, and for the public documents a separate public database can be defined.

Mask characters

Several numeric and alphanumeric characters can be used to mask the fileid. These characters can be:

1. '#' : To identify one numeric character
2. '@' : To identify one alphabetic character
3. '?' : To identify one alphanumeric character
4. '*' : To identify up to 8 alphanumeric characters (as a wildcard)

Also it is possible to use constants.

For example, for a database that contains only the file with filename starting with the string 'ABC' and has an undetermined filetype; the mask should be

ABC????? *

The Document Files

The document files are the files that constitute the database. These files must have a specific format. First of all they must have a header part, a body part and the banner separating the two parts. Body part contains the text of the document. The header part is used to determine the specific search fields in the document. These fields are:

Title
Author(s)
Date
Committee
Document
Revision
Supersedes
Status
Maintainer
Access

The description and regulations about these keywords are explained in the file EXEC106 89 - Guidelines for the Production of EARN Documents by P. Bryant and H. Nussbacher.

If one or more of the keywords are missing, then the driver replaces it with '???' string.

LSVDBEAR DRIVER

The basic commands of the driver are:

SYN : Makes the relationship with the keyword code stored in the DBINDEX file and its name entered by the user.

PRINT : Selects the right procedure to print the requested report. This procedure returns the text to print to LSVDBASE.

PRINTEXL : Gives the list of portions which are in fact report, ie LSVDBASE must call the driver for printing.

INDEX : Defines the default format used for the INDEX command : column contain and width, justification (left (L), right (R), centered (C) or right with leading zeroes (R0)) and headings.

OPEN : Checks the availability of the requested database and its up-to-date. every time a session is opened. If the conditions for automatic set up are matched, the database is set up again.

REFRESH : Forces the database update, only used either by the Listserv postmaster or Listserv itself. This procedure contains the statements to perform when the database is updated : at least the DBNAMES and the DBINDEX removal.

CLOSE Not used.

LIST Prepares a short description for the database. This command is issued by Listserv when a user asks for the database list.

For more information and examples regarding the commands the document LISTDBC MEMO written by Nadine Grange is suggested.

ASTRA-LISTSERV Databases Access

The ASTRA service offers the possibility to access to the EXEC and BOD documentation directly though the User Interface. The EXEC and BOD databases are accessible as any other ASTRA database, using the interface facility.
These requests are sent via mail to LISTSERV, and the responses arrive also via mail.
In the future it will be possible to access also to other EARN documentation through ASTRA interface.

For example if you want to search the EXEC documents containing the word gateway, you must write:

Current database is: EXEC
Search about gateway

If you want to receive only the index of these documents, you must select the NUMERICAL result, if you prefer to receive the whole documents, you must select the DOCUMENTS result.
If you write:

Current database is: EXEC
Search about gateway
Send me the NUMERICAL results of this query on the selected DB (X)

You will receive:

```
> SEARCH ( ( GATEWAY ) ) in EXEC
--> Database EXEC, 68 hits.

> Index
Ref#   Date     Document              Title
----   ----     --------              -----
0002 89/??/?? EXEC28  89
0018 89/??/?? EXEC72  89
0028 89/??/?? EXEC67  89
0035 89/??/?? EXEC99  89
0037 89/??/?? EXEC95  89
0038 89/??/?? EXEC94  89
0043 89/??/?? EXEC97  89
0044 89/??/?? EXEC96  89
0048 89/08/21 EXEC119  89      EARN project definition
                               group ASTRA - proposal
0049 89/08/21 EXEC118  89      Proposal for an EARN
                               project group on value add+
0053 89/08/17 EXEC111  89      EARN traffic and topology
0054 89/08/17 EXEC108  89      NOG meeting May 25-26 1989
.
.
.
.
.
```

If you write:

Current database is: EXEC
Search about gateway
Send me all the DOCUMENTS wich satisfy this query on current DB (X)

You will receive:

```
LISTSERV at UKACRL     EXEC28 89 EARN-MIN
pb222                                       EXEC/28/89
                                      update of EXEC/7/89
EARN EXEC

Revised OSI transition plan
Approved by the EARN Executive February 23, 1989
                                             issued by
                                             D Jennings
                                            March 3, 1989

---------------------------------------------------------
Although the OSI Transition is progressing rather slowly,
it is progressing.
.
.
.
.
.
```

Moreover if a user prefers to write a query with the LISTSERV language, he can use the LDBASE facility (PF12) directly from the User Interface.
In this case if you want to search the EXEC documents containing the word gateway, you must write:

Search gateway in EXEC

For more informations about the LISTSERV database language see the file LISTDB MEMO.
From the Extended Features screen (PF4), it is possible to send more complex queries using the Aquarius Language:
in the field Query #1,..Query #4 you can define the words with which the research will be made in the database.
For example if you want every document dealing with TCP IP you must write:

Query #1: TCP IP

This query is translated in: "Search TCP IP in EXEC"

On the contrary if you want every document, which title contains the words TCP IP you must write:

Query #1: TCP IP.title.

This query is translated in: "Search * in EXEC where title contains TCP IP"

Moreover if you want to receive the documents you can select the option DOCUMENT, if you want to receive the whole documents (X) or some specific field (name of the fields).
For example if you want receive only the field Author of the documents that deals with TCP IP you must write:

Query #1: TCP IP
DOCUMENT.......: author

This query is translated in: "Search TCP IP in EXEC";"Print author" and you will receive:

```
> SEARCH ( ( TCP IP ) ) in EXEC
--> Database EXEC, 37 hits.

> print AUTHOR
>>> Item number 37, dated 89 -- AUTHOR
???

>>> Item number 39, dated 89 -- AUTHOR
???

>>> Item number 43, dated 89 -- AUTHOR
???

>>> Item number 54, dated 89/08/17 -- AUTHOR
T KALFAOGLU

>>> Item number 55, dated 89/08/17 -- AUTHOR
VARIOUS
.
.
.
.
.
```

ASTRA Proposal for EARN Documentation

Introduction

The LISTSERV database feature was designed to let the users access the files on LISTSERV gathered as a database. The facility is only available for LISTSERV lists and the NOTEBOOK, BITEARN, and PEERS files. A new driver is developed to access other database files. The driver is called LSVDBEAR and it uses the LISTSERV Database capability and allows to define and to search a more general kind of database.

This document proposes the four major points of evolution for the LISTSERV - EARN database. Also it defines a common way to access files stored under LISTSERV via the common ASTRA User Interface.

The major points of the proposal are:

* Meta-language
* EARN documentation
* Local documentation
* Logical databases

Meta-language

A proposal for having a more general driver working with the EARN database is to introduce a meta-language to use in describing, in a standard format, the files contained in a database. A database groups many files on the base of their arguments but does not check the characteristics of these files. So it is not important for the database to have many different format of the files. A standard meta-query-language can be developed and also all the files in the databases can be stored in a standard format.

The most important part of the document is the header, because the header contains the keywords about the document itself and in most cases the search is performed on the header.

Example:

The most general and accepted header format for the EXEC and BOD documents is the one proposed by Paul Bryant and Hank Nussbacher. The details of this proposal is explained in the executive document EXEC106 89. Many files created before this proposal do not respect to these standards. When the driver accesses those files, it cannot find the keywords it is looking for, so it substitutes the dates with '?' and leaves TITLE field blank.

The result of a search from a document with the correct format:

```
Ref#  Date      Document       Title
----------------------------------------------------------
0002  89/08/21  EXEC118 89     Proposal for an EARN project group on value added services
```

The result of a search from a document with another format:

```
Ref#  Date      Document       Title
----------------------------------------------------------
0004  89/??/??  EXEC119 89
```

It can easily be seen that only some parts of the header is necessary to save time and effort. As a proposal it can be suggested that title, date and author fields of a document should be in a defined format, and can be present for every kind of document. Moreover, for each group of documents also some specific field describing the characteristics of the group, should be in a defined format.

Also the body and the banner are the two as important fields as the header of the document. It must be known if they are present and if so, what kind of characters compose the banner.

The use of a carefully structured meta language can make the driver more portable and the searches will be easier.

EARN Documentation

There are many manuals and documents written about EARN. These mostly deal with the services provided, products available or general information. These are scattered over LISTSERV and some on NETSERV. When a user wants to access an EARN document, he must at first, find out the location of that document. For a new user this can be a problem. Another problem is the impossibility of searching these documents by keywords.

ASTRA proposes to put the EARN documents under LISTSERV and make them accessible through the LSVDBEAR driver by the help of the ASTRA User Interface. Each EARN database will be a group of EARN documents dealing with the same arguments.

In this way all EARN documentation i.e the minutes of the meetings, the proposals, the announcements and the guide and the manuals will be available to the users via the ASTRA User Interface. The installation of a database under ASTRA is an easy task (see LISTDBED MEMO).

Example:

EARN documentation contains the information about EARN services. A database could be installed on Listserv named EARNSERV, which will group every document dealing with "EARN Service". When a user would search some information about a specified service, he could send a simple query to Listserv or, still easier, use the ASTRA user interface. A search that actually is very hard and which could be unsuccessful, could be successfully made in a little time and without any effort.

Local Documentation

In every EARN node there are many documents prepared by the local staff.These documents can be guidelines for a specific service, manuals for some products, minutes of local meetings or descriptions of locally developed software or services. These documents can be either public or private. By grouping these documents as a database, all the users can access them through the common ASTRA User Interface. These documents can be in any language desired. The access to them can be controlled. For example, some documents may only be accessed by the local users or some part of the local staff whereas the others can be made publicly available.

To accomplish this, on one node in every country, an ASTRA server can be installed. Only one or two people are enough for the server maintenance and the installation of databases and documents.

The local documents should be stored in the meta-language format used by all of the servers. For the installation of the databases and the privacy control see the document LISTDBED MEMO.

Logical Databases

A facility offered by the ASTRA service is to group a collection of databases dealing with common arguments in a single database. When a user sends a query on this logical database, this request is executed on any database grouped under it. In this way a user should not send the same query on more databases, but this is automatically executed by the ASTRA service.

This facility could be a further help to access the EARN documentation. Infact this documentation could be structured in specific databases, which, in turn, will be grouped in logical databases.

Example:

The EARN documentation about the EARN service can be splitted in many databases, on the ground of the specific service they describe. A database about ASTRA database service, another about the Listserv database service, another about the Listserv mailing-list service, etc. can be made. A logical Listserv database could be a collection of each one of these databases about Listserv service. Another logical database could be a logical database containing the databases dealing with the "database service": ASTRA service, Listserv service, etc. When a user want to search some general information about Listserv or about database service, he can send a request to the logical database "LISTSERV" or "DBSERV" and his request is executed on any database belonging to LISTSERV ot to DBSERV.

As an example to this, a database named EARN can be described. This database consists of two other logical databases EXECDOC and BODDOC. EXECDOC contains two databases EXEC and EXECPRIV and BODDOC contains BOD and BODPRIV. When a user sends a query to EARN this is sent to all of those databases if the user has the right to access them. If the user does not have the right to access the private documents then the query will not be sent to EXECPRIV and BODPRIV documents.

LSVDBEAR LISTSERV driver: functional specifications.

The existing LISTSERV database functions let the users access the information stored in different way using different access programs (drivers) as follows:

- NOTEBOOK files using LSVDBNB driver
- LIST files using LSVDBLS driver
- BITEARN NODES file using LSVDBBN driver
- PEERS file using LSVDBPN driver

To manage different types of documentation such as EARN documents or something else a new driver has been implemented. The driver name is LSVDBEAR and has been written in REXX Language. The new driver allows every type of lists of files to be used as a database.

A LISTSERV database is composed by the whole document files belonging to the database and two files to query and to print it.

- Any document file is divided by a banner into two parts.The banner can be defined in the LSVDBEAR DBTABLE. The LSVDBEAR table is fully described in the document ASTRA LISTDBED.
  - The header part which contains some general infomation about the document.
  - The body part, is really the document stored on the database.

  For more details about the field defined in the header and the general document structure see "Guidelines for the Production of EARN Documents" by P. Bryant and H. Nussbacher (EXEC106 89).

- The <dbname> DBNAMES file contains an entry for any document belonging to the database. Any entry contains:
  - The DDNAME which connects any entry in the DBNAMES file to the corresponding entry in the DBINDEX file.
  - The file identification of the file which contains the document.
  - The number of records of the file.
- The <dbname> DBINDEX file contains the values of the field gotten from the header part of each document and some information about the location of the header and the start position of the body.

Anyway, for more details about the DBINDEX and DBNAMES file structuıe, please, see the document LSVDBC MEMO written by Nadine Grange.

This driver is called by the LSVDBASE main procedure. The calling list is composed by 2 types of parameters: the request type and the parmlist where the <DBname> and the other values are specified. The following is the complete list of the available request types.

```
When reqtype == 'SYN' Then Signal Synonym
When reqtype == 'OPEN' Then Signal Open
When reqtype == 'CLOSE' Then Exit 0
When reqtype == 'INDEX' Then Signal Index
When reqtype == 'REFRESH' Then Signal Refresh
When reqtype == 'PRINTEXL' Then Signal Print_EXL
When reqtype == 'PRINT' Then Signal Print
When reqtype == 'LIST' Then Signal List
```

If the request is INDEX the driver returns a collection of data about the main fields of the request database. The fields will be printed for any document selected by the search/select command. These main fields are defined by the people who had implemented the driver.

```
The  EXEC  database (The database about the EARN Executive
public documentation) uses  the  Reference,   Date,
Document and Title fields.

> Index
Ref#  Date   Document          Title
----  ----   --------          -----
0048 89/08/21 EXEC119 89      EARN project definition
                              group ASTRA - proposal
0049 89/08/21 EXEC118 89      Proposal for an EARN
                              project group on value add+
```

If the request is SYNONYM the driver returns the key to identify the fields stored in the <dbname> DBINDEX file. Infact in the <DBname> DBINDEX file any field is identified by one alphabetic character associated to a more significative keywords the users use to query the database.

If the request is REFRESH the driver controls if the user doing the request is allowed to refresh the database. Then the <DBname> DBNAMES and the <DBname> DBINDEX files are erased.

If the request is LIST the user wants to know the available databases in the system. The driver controls the access list parameter in the <DBname> LIST file and if the user is allowed it returns a brief description about the database.

If the request is OPEN the driver controls if the user can access the database. When the OPEN function is called the, database can be in two different states. It can need to be upgraded or it can be ready to be queried.

- If no upgrade is to be done, the driver must only control if the user is allowed to access the database. If it is possible, the database is accessed and the results will be returned to the user.
- If some upgrade must be done the driver should decide if new files can be inserted in the database. The driver looks to the file LSVDBEAR DBTABLE for the mask value, the banner and the associated access file.
    - The mask value defines some restrictions on the filename and the filetype of the files to be loaded in the database. The mask is used to verify if the filename and the filetype of the file to be loaded match the naming convention defined against the files belonging to the database.
      For example the mask for the EXEC database is EXEC#### ## and this implies that only the files with a filename composed by the fixed part EXEC followed by up to 4 numbers and a filetype composed by up to 2 numbers can be accepted.
    - The banner is used at least as a boundary between the header and the body part.
    - The access file is the file which contains the filename and the filetype of any file belonging to the database. Any entry in the access file contains the associated FAC description. If the FAC description of the entry in the access file matches the general FAC description of the database defined in the LSVDBEAR DBTABLE the file is loaded in the database and the DBNAMES and DBINDEX files are upgraded.

If the request is PRINT the selected entry will be printed according to the user requirements.

The PRINTEXL, and CLOSE requests will be not discussed.

The keywords of a database (the value of the fields loaded in the DBINDEX file) are defined according to the standard document description - Guidelines for the Production of EARN Documents by P. Bryant and H. Nussbacher (EXEC106 89).

If some users want to change the keyword names or change the keywords some changes should be done to the LSVDBEAR driver:

1. In the SYNONYM subroutine change the name and the qualifier of the keywords according to the requirements.
2. In the INDEX procedure change the field to be displayed according to the user requirements.

3. In the PARSE_HEADER procedure change the keyword names according to the change done in the SYNONYM procedure.

4. In the NEW_ITEMS procedure change the keyword default values according to the change done in the SYNONYM procedure.

For more details see: LSVDBNK MEMO document.

The changes to the LSVDBEAR driver could be done as in the following example:

A subset of the original SYNONYM procedure

```
......
.......
Select
 When k & Abbrev('TITLE',name,1) Then Exit 'T'
 When k & Abbrev('AUTHOR',name,2) | Abbrev('AUTHOR(S)',name,2) Then Exit 'A'
 When k & Abbrev('DATE',name,1) Then Exit 'D'
 When k & Abbrev('NAME',name,1) | Abbrev('DOCUMENT',name,1) Then Exit 'N'
 When k & Abbrev('STATUS',name,1) Then Exit 'S'
 When k & Abbrev('MAINTAINER',name,1) Then Exit 'M'
 When k & Abbrev('ACCESS',name,1) Then Exit 'X'
 When ¬p Then Exit ''
 /* Portions only */
 When Abbrev('BODY',name,1) | Abbrev('TEXT',name,1) Then Exit 'B'
 When Abbrev('HEADER',name,1) | name == 'HDR' Then Exit 'H'
 Otherwise Exit ''
End
```

A subset of the original PARSE_HEADER procedure

```
......
.......
If Find('DATE AUTHOR(S) AUTHOR TITLE STATUS DOCUMENT ACCESS MAINTAINER',kwd),
 ¬== 0 Then h.kwd = data
   field = line
End
Parse var field kwd .':' data
Upper kwd
If Find('DATE AUTHOR(S) AUTHOR TITLE STATUS DOCUMENT ACCESS MAINTAINER',kwd),
 ¬== 0 Then h.kwd = data
$ = 'TITLE'; title = Space(h.$)
$ = 'AUTHOR(S)'; authors = Space(Translate(h.$))
$ = 'AUTHOR'; author = Space(Translate(h.$))
$ = 'DOCUMENT'; document = Word(Space(h.$),1)||' '||Word(Space(h.$),2)
$ = 'STATUS'; status = Space(h.$)
$ = 'MAINTAINER'; maintainer = Space(Translate(h.$))
$ = 'ACCESS'; access = Space(h.$)
$ = 'DATE'; date = Space(h.$)
Return
```

A subset of the original NEW_ITEMS procedure.

```
......
......
If title = '' Then FT = ''
                Else FT = '15'x||'FT' title
......
......
Else date = ftn||'??'||'??'
Call Check 'LSVFWR1' fn 'DBINDEX' fm dd strecno nrecs.fileid,
date'15'x||'FA' author||FT'15'x||'FN' document'15'x||,
'FS' status'15'x||'FM' maintainer'15'x||'FX' access||,
PB'15'x||'PH 1' body-2
......
```

The new version of the SYNONYM, NEW_ITEMS and PARSE_HEADER procedure contain the following changes:

- The keywords "TITLE" has been removed.
- The keywords "AUTHOR" and "AUTHOR(S)" has been changed to "AUTORE" and "AUTORI".
- The "EXPIRATION" keywords has been added using the prefix "E".

A subset of the new SYNONYM procedure

```
......
.......
Select
 When k & Abbrev('EXPIRATION',name,2) Then Exit 'E'
 When k & Abbrev('AUTORE',name,2) | Abbrev('AUTORI',name,2) Then Exit 'A'
 When k & Abbrev('DATE',name,1) Then Exit 'D'
 When k & Abbrev('NAME',name,1) | Abbrev('DOCUMENT',name,1) Then Exit 'N'
 When k & Abbrev('STATUS',name,1) Then Exit 'S'
 When k & Abbrev('MAINTAINER',name,1) Then Exit 'M'
 When k & Abbrev('ACCESS',name,1) Then Exit 'X'
 When ¬p Then Exit ''
 /* Portions only */
 When Abbrev('BODY',name,1) | Abbrev('TEXT',name,1) Then Exit 'B'
 When Abbrev('HEADER',name,1) | name == 'HDR' Then Exit 'H'
 Otherwise Exit ''
End
```

A subset of the new PARSE_HEADER procedure

```
......
.......
If Find('DATE AUTORE AUTORI EXPIRATION STATUS DOCUMENT ACCESS MAINTAINER',kwd),
 ¬== 0 Then h.kwd = data
   field = line
End
Parse var field kwd .':' data
Upper kwd
If Find('DATE AUTORE AUTORI EXPIRATION STATUS DOCUMENT ACCESS MAINTAINER',kwd),
 ¬== 0 Then h.kwd = data
$ = 'EXPIRATION'; expiration = Space(h.$)
$ = 'AUTORI'; authors = Space(Translate(h.$))
$ = 'AUTORE'; author = Space(Translate(h.$))
$ = 'DOCUMENT'; document = Word(Space(h.$),1)||' '||Word(Space(h.$),2)
$ = 'STATUS'; status = Space(h.$)
$ = 'MAINTAINER'; maintainer = Space(Translate(h.$))
$ = 'ACCESS'; access = Space(h.$)
$ = 'DATE'; date = Space(h.$)
Return
```

A subset of the new NEW_ITEMS procedure.

```
......
......
If expiration = '' Then expiration ='???'
......
......
Else date = ftn||'??'||'??'
Call Check 'LSVFWR1' fn 'DBINDEX' fm dd strecno nrecs.fileid,
date'15'x||'FA' author'15'x||'FE' expiration'15'x||,
'FN' document'15'x||,
'FS' status'15'x||'FM' maintainer'15'x||'FX' access||,
PB'15'x||'PH 1' body-2
......
```

References:

- LISTSERV database function (E.Thomas) document LISTDB MEMO.
- LISTSERV database creation (N.Grange) document LISTDBC MEMO.
- Guidelines for the Production of EARN (P.Bryant, H.Nussbacher) document EXEC106 89.
- ASTRA-LISTSERV database installation (E.Delen, S.Giordano, G.A.Romano, D.Vannozzi) document ASTRA LSVDBED.
- ASTRA-LISTSERV database access (E.Delen, S.Giordano, G.A.Romano, D.Vannozzi) document ASTRALSV ACCESS.
- ASTRA-LISTSERV database description (E.Delen, S.Giordano, G.A.Romano, D.Vannozzi) document ASTRALSV DESCRIPT.
- ASTRA-LISTSERV keyword changing (E.Delen, S.Giordano, G.A.Romano, D.Vannozzi) document ASTRALSV LSVDBNK.

ASTRA-LISTSERV driver: how to change the keywords.

The LISTSERV driver named LSVDBEAR allows every type of lists of files to be used as a database. This driver is called from the LSVDBASE procedure, when a request for a database < DBname > arrives. The keywords used for the queries to a database are not decided a priori. For the EXEC BOD database the keywords have been chosen according to the EARN document description - Guidelines for the Production of EARN Documents by P. Bryant and H. Nussbacher (EXEC106 89). The procedure to change the driver in accordance with the specification of new keywords is the following.

Step 1

Change the 'SYNONYM' subprocedure. This procedure translates the keywords used by the users in the corresponding keywords used in the < DBname > DBINDEX file. The letters "K" and "P" are used to indicate if the request is to search a keyword or if it is to search a position.

The forms to use for this translation are:

- When k & Abbrev(< keyword > ,info,lenght) Then Exit < sfk >
- When p & Abbrev(< position > ,info,lenght) Then Exit < sfp >
- Where < keyword > and < position > are the keyword and the position-word used by the users and < sfk > and < sfp > are the short form of keyword and position-word used in the < DBname > DBINDEX file.

This process must be done for each keyword and each position.
It would be a good trick to use as keywords the real keywords used in the header of the files.
This real keywords must appear in the followed form:

```
Header
  RK1: Info1
  RK2: Info2
  .
  .
  RKn: Infon
```

Step 2

Change the 'PARSE_HEADER' subprocedure. This procedure is used to search the keywords used in the files header. When a keyword is found its data are associated with a variable through the followed commands:

```
If Find( RK1 RK2..RKn,kwd) ¬= 0 Then h.kwd = data
$ = RK1 ; var1 = Space(h.$)
$ = RK2 ; var2 = Space(h.$)
.
.
$ = RKn ; varn = Space(h.$)
```

Step 3

Change the 'NEW_ITEMS' subprocedure. This procedure is used to write on the < DBname > DBINDEX the lines related with each file. When a variable is found empty it could be a good trick to insert some question mark to show to the user that the field has been not found instead to return an empty line.

ASTRA-LISTSERV database: an example.

The LSVDBEAR dbtable at CNUCE

```
DBname FN FT BannerChar BannerLength Fac Factablen Factablet
EXEC EXEC####    ## - 50 ALL
BOD BOD#####  ## - 50 ALL
EXECPRIV EXEC####    ## - 50 PRV  EXEC ACCESS
BODPRIV BOD#####  ## - 50 PRV  BOD ACCESS
DUMMY  *  NOTE?       = 50 PRV  DUMMY ACCESS
ASTRA  ASTRA?   *     - 50 ALL  ASTRA ACCESS
```

The ASTRA public documentation ACCESS file

```
ASTRA     INET91    ALL
ASTRA     LISTDBED  ALL
ASTRADB   SURVEY    ALL
ASTRALSV  ACCESS    ALL
ASTRALSV  DESCRIPT  ALL
ASTRALSV  LSVDBEAR  ALL
ASTRALSV  LSVDBNK   ALL
ASTRALSV  PROPOSAL  ALL
```

The ASTRA public documentation LIST file

```
*
*  Earn test ASTRA Database
*
*  REVIEW= PUBLIC          SUBSCRIPTION= CLOSED      SEND= private
*  NOTIFY= YES             REPLY-TO= LIST,RESPECT    FILES= YES
*  VALIDATE= all commands  STATS=Extented,owner        ACK= NO
*  CONFIDENTIAL= NO        Loopcheck= Notcount       Renewal= none
*  NOTEBOOK= NO,A,single,public    Errors-to= Owner
*  formcheck= no
*  OWNER= ESRA@ICNUCEVM
*  OWNER= (ASTRA-WG@ICNUCEVM)
*
*
*
*
* PW= ESRA
```

The ASTRA public documentation FILELIST file

```
* ASTRA FILELIST  for LISTSERV at ICNUCEVM.
*
*  ASTRA Filelist
*
*  This file lists all files related to ASTRA
...........
...........
* ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
*
*  The GET/PUT authorization codes shown with each file entry describe
*  who is authorized to GET or PUT the file:
*
*
*+    LAS = 'ASTRA@ICNUCEVM'      /* */
*+    AWG = (ASTRA-WG@ICNUCEVM)   /* ASTRA Working Group */
*+    APG = (ASTRA-PG@ICNUCEVM)   /* ASTRA Project Group */
*
*
* ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

*
* Filelist entries:
*
*                  f         rec                  last - change
* filename filetype m GET PUT -fm lrecl nrecs   date     time
* -------- -------- - --- --- --- ----- ----- -------- --------


ASTRALSV DESCRIPT   ALL AWG V      68   191 91/06/26 13:42:54
ASTRALSV ACCESS     ALL AWG V      73   174 91/06/26 13:44:00
ASTRA    LISTDBED   ALL AWG V      72   475 91/06/26 13:45:38
ASTRALSV PROPOSAL   ALL AWG V      68   203 91/06/26 13:46:29
ASTRADB  SURVEY     ALL AWG V      68   243 91/06/26 13:47:23
ASTRA    INET91     ALL AWG F      80   258 91/06/26 13:48:15
```