

Database  
Management  
System

**LINTER**®  
Version 5.9

## Backup and Restore

Relational Expert Systems

---



# Table of contents

<b>DBSTORE</b> .....	<b>5</b>
Hardware and Linter Hardware Requirements.....	5
Launching DBSTORE .....	5
Command Line Options .....	5
Unloading a DB .....	8
Restoring a DB .....	8
<b>LHB</b> .....	<b>10</b>
Overview .....	10
lhb Command, Options, and Arguments .....	10
Mode Options .....	12
Software Control Options.....	13
Object Description Options .....	14
Incremental Archiving Options.....	15
Priority options Option.....	16
Control point options Option .....	16
Tape Backup Options .....	16
Error Handling Options .....	16
Launching lhb .....	17
User Interface Language.....	17
Backup .....	17
Restore .....	18
Working with Archives .....	19
Incremental Archiving.....	19
Archiving on Tape .....	20
Archive Status .....	20
Archive Testing.....	21
Automating Backup .....	22
Script Structure.....	22
Set Time and Date Format .....	22
Variables.....	24
Scheduling Execution .....	24
Hourly .....	24
Daily .....	25
Weekly .....	25
Monthly.....	25
Specified Time .....	26
Special Operations.....	26
Actions on Startup .....	27
Actions on Termination.....	27
Error Handling Actions.....	27
Scripting Operators, Comments, and Statements .....	28
Operators .....	28
Comments.....	28

---

Scripting Statements .....	28
Loops .....	28
Conditional .....	29
Backup .....	29
Copy File .....	29
Move File .....	30
Rename File .....	30
Delete File .....	30
Create Directory .....	31
Launch Program .....	31
Execute Program .....	31
Exit lhb .....	31
Break .....	31
Exception Handling .....	31
Error Handling .....	32
Predefined Variables and Constants .....	32
Predefined Functions .....	32
Sample Script .....	33
Backup of a Working DB .....	34
lhb Messages .....	34
Execution Report .....	34
Common Messages, Causes, and Suggestions .....	35
Termination Codes .....	36

## Introduction

To eliminate the chance of accidental or malicious loss of data, some of which may have been gathered over many years, DB administration must include regular, complete or selective backup. The best practice suggests at least two copies on separate magnetic media stored off-site in separate locations.

Linter provides two utilities for backup and restoration of databases `dbstore` and `lhb`. Both operate from the command line.

`dbstore` is primarily used to generate ASCII format backup for later transfer to another DB.

`lhb` is a comprehensive, Linter DB format backup system that provides for a variety of types of backup and scheduling automated backup.

## DBSTORE

The dbstore program is used to back up tables to .sql and .lod text files. It is launched from the command line of every OS to which Linter has been ported.

### Hardware and Linter Hardware Requirements

Before running dbstore, the following must exist

- 1) Enough free hard drive space for the data being unloaded;
- 2) An active Linter kernel;
- 3) Six free channels;
- 4) The user must have DBA access privileges;
- 5) To back up the entire DB, every DB user must have provided at least SELECT access rights on their respective tables to the DBA or password.

### Launching DBSTORE

The dbstore program is launched from your operating system's command line. The program executable is dbstore.exe for Windows and dbstore for UNIX.

The dbstore command accepts several options, and their arguments, described in the following section.

The command, with its options, and arguments maybe entered on the command line. Alternatively,

dbstore <path>,  
may be entered on the command line.

Here, <path> – is the full path to a text file containing the complete dbstore command, including options and arguments. If <path> does not begin at the root directory, the search will begin in the current working directory.

The complete command with all its options and arguments, may not contain a new line character.

dbstore evaluates only one command line each time it is launched.

### Command Line Options

The command and its options are:

```
dbstore [-d | -ds| -ex| -ep| -f| -g| -h| -l| -n | -o| -oa | -op
| -oq| -or| -os| -ostruct| -ot| -otrig| -ou| -ov| -p| -plist| -q
| - r| -s| -t| -u| -version| -xml| -z| ]
```

Comments:

- 1) Each option requires one or more arguments, for some of which defaults or prompts are provided.
- 2) The options may be entered in any order.
- 3) The command dbstore and its options may be entered in lower or upper case.
- 4) Identifiers such as table names and passwords are case sensitive.

- 5) Each options is preceded by a minus, -, sign. The alternative back slash, \, may be used in all operating systems except Unix.
- 6) A - or / followed by a character that is not an option character is ignored and program is terminated.
- 7) The command dbstore followed by the question mark,?, will display a list of the options available to the command.

The following table describes each of the options

<b><u>Option</u></b>	<b><u>Argument</u></b>	<b><u>Description</u></b>
-d	path	Path defines the path to the directory where all the files created by dbstore will be saved.
-ds		Set formatting for the table and column names in the produced SQL-file by addition extra spaces to make all names the same length (18 characters).
-ex		Disable unloading EXTERN files.
-ep		Set using the blank user's password and disable prompt for password..
-f	File_name	Set the name of the file where all data will be saved.
-g		Set converting BYTE and VARBYTE columns to character string during unloading process.
-h		Display a list of the options available to the command.
-l	table_name	Unload only data from specified table. SQL-files are not created.
-n	server name	Server_name identifies a remote Linter server if the table(s) being unloaded are located there.
-o	argument[...]	<p>The -o option may be followed by a space-separated list of single-character codes. These character codes must be capital letters.</p> <p>B During a subsequent restore, the default GENDB settings will be used.</p> <p>D «fast load» option will be added to the loarel data load program.</p> <p>F Unload all BLOB objects into a single file.</p> <p>M Use the GETM command.</p> <p>S Backup the DB structure only. Data is not saved.</p> <p>T Overrides default of not creating backup tables if originals are empty. This argument will create a backup table with 1 row for each empty table.</p>
-oa		Set unloading for all access rights.
-op	procrdure_name[,...]	Set the list of the stored procedures to be unloaded. The rules of using this option are the same as those for the -ot option.
-oq	sequence_name[,...]	Set the list of the sequences to be unloaded. The rules of using this option are the same as those for the -ot option.
-or	role_name[,...]	Set the list of the roles to be unloaded. The rules of using this option are the same as those for the -ot option.

<u>Option</u>	<u>Argument</u>	<u>Description</u>
-os	synonym_name[,...]	Set the list of the synonyms to be unloaded. The rules of using this option are the same as those for the -ot option.
-ostruct		Set unloading for DB structure.
-ot	table_name[,...]	The same as -t option (made for compatibility of the versions).
-otrig	trigger_name[,...]	Set the list of the triggers to be unloaded. The rules of using this option are the same as those for the -ot option.
-ou	user_name[,...]	Set the list of the users to be unloaded. The rules of using this option are the same as those for the -ot option.
-ov	view_name[,...]	Set the list of the views to be unloaded. The rules of using this option are the same as those for the -ot option.
-p	null_char	Specify null character. By default, it is NULL.
-plist		Defines the complete specification of the text file containing a list of names and passwords of the users of the unloaded DB in following format <user_name>/<user_password>. This file is created in advance (before dbstore is started).
-q		Disable all questions
-r	reserved_pages	Specify the size of the table during unloading by the loarel program. The value of reserved_pages is a positive integer equal to the number of pages. 1 page is 4096 bytes. This option is useful when using dbstore to merge tables or append a large amount of data to an existing table; i.e., situations when the resulting table file will be larger than the original.
-s		Set unloading SQL-queries for creation references after SQL-queries for creation tables in the same produced SQL-file.
-t	table_name[,...]	The option is followed by a comma-separated list of tables to be unloaded. The following rules apply to each table name: <ol style="list-style-type: none"> <li>1 It must correspond to a user table. System tables can not be unloaded.</li> <li>2 It must not contain the owner's username. If the DB has two or more tables with the same name but different owners, all those tables will be unloaded</li> <li>3 It may be defined using the SQL language construct % and _ for wildcards.</li> <li>4 If a table is empty, no backup file is created. This default may be overridden with the T argument to the -o option.</li> </ol>
-u	user_name/password	If this option is omitted, dbstore will prompt for both name and password after the program starts.
-version		Display information of the version of the dbstore.
-xml		Unloading data in XML format.
-z		Set replacing '\$' symbol to '\\$' symbol. (Usually for UNIX systems).



If none of the above keys are set, the entire database is unloaded.

## Unloading a DB

When unloading a DB, all files created in the process are located in the directory set by the `-d` option. If that option is not used, the files will be located in the sub-directory `DB_STORE`, located in the directory that contains the `dbstore` program. Usually, `dbstore` is located in `~/Linter/bin`.

During the download process, new sub-directories and batch files are created. A sub-directory is created for each DB object owner. All user data is unloaded into the corresponding user's directory. The user subdirectories are named `<user_name>.lod`. Filenames in each directory are created as `<object_name>.lod`. In case that the filename is specified `<object_name>.lod` files are not created.

The `DB_STORE` directory contains the following files:

<code>crdb.gdb</code>	Text for CREATE DATABASE.
<code>create_0.bat</code>	Unload DB creation file for use by <code>gendb</code> to create DB.
<code>create_1.bat</code>	System table creation file for use by <code>inl</code> to create DB dictionary.
<code>create_2.bat</code>	DB loading file.
<code>acc_0001.sql</code>	Access rights creation file.
<code>a1users.sql</code>	Change user password file.
<code>cr_users.sql</code>	User creation file.
<code>idx_0001.sql</code>	Index creation file.
<code>syn_0001.sql</code>	Synonym creation file.
<code>tab_0001.sql</code>	User table creation file.
<code>trig0001.sql</code>	Trigger creation file.
<code>proc0001.sql</code>	Procedure creation file.
<code>view0001.sql</code>	View creation file.
<code>rol_0001.sql</code>	Role creation file.
<code>ref_0001.sql</code>	Reference creation file.
<code>seg_0001.sql</code>	Sequence creation file.

## Restoring a DB

You restore a DB previously unloaded using the `dbstore` program using the following steps:

1. Run the `create_0.bat` file created during the unloading process. This file contains a call to `gendb` with the parameters defined by the `dbstore` program in the `crdb.gdb` file. (The `gendb` program version and Linter version must be the same.)
2. Launch the Linter kernel.

3. Run the create\_1 .bat file created during the unloading process. This file contains a call to the inl program to generate system objects.
4. Run the create\_2.bat file created during the unloading process. This file contains calls to the inl and loarel programs. They create all users, with empty passwords, tables, views, synonyms, roles, access rights, triggers, and procedures.
5. Enter all the user passwords using ldbal or inl.

## LHB

lhb supports the command line interface of all operating systems on which Linter runs. For access to the full range of lhb options, the user must be of type DBA.

## Overview

lhb provides the following backup functions and options:

- 1) Complete or selective DB backup;
- 2) Automated backup triggered by time and or other criteria;
- 3) Complete or selective DB restoration, even with an inactive kernel;
- 4) Backup during multi-user session;
- 5) Backup and restoration of:
  - table structures,
  - user profiles,
  - table data,
  - roles and role assignments,
  - views,
  - access rights,
  - synonyms,
  - link integrity;
- 6) Saving selected user objects;
- 7) Automatic definition of dependent objects;
- 8) Archive functions:
  - list and test or integrity,
  - secure with password,
  - increment, expand,
  - multi-volume.

## lhb Command, Options, and Arguments

With no options or arguments, lhb returns a list of its options and arguments including a brief description of each.

Where the modified BNF structure of the lhb command includes space separated lists of options, one or more of such options may be included in the command

```
lhb ::= [ <Mode options> [ ... ] [ <Option > <Argument> [, ... ] ]
```

```
<Mode options> ::=
  { cp | ef | l | r | s | t | tape | script | set }
  [ set<message_language>]
```

```
<message_language> ::= { russi an | engl ish }
```

```
<Option> ::=
  [<Software control>|<Object description>
  |<Incremental archive>
  |<Priority>
  |<Tape backup>
  |<Error handling>]
```

```
< Software control > ::=
  [{-lr | -le}]
  | -ef
  | -pg
  | -c<"comment">
  | -f<archive_file>
  | -ft<script_file>
  | -fl <script_file>
  | -g<archive_password>
  | -n<server_name>
  | -out<msg_file>
  | -out+<msg_file>
  | -p<path_to_restore>
  -u<uname/password> -v[<size>[ K M ] ]
```

<"comment"> ::= user comment saved in archive file

<archive\_file> ::= path, including name, of archive file

<script\_file> ::= path, including name, of backup script

<archive\_password> ::=

if set, password is required to restore DB

<server\_name> ::= node name of remote server where DB located

<msg\_file> ::=

path, including name, of file to hold lhb messages

<path\_to\_restore> ::=

path to directory in which DB will be restored

<uname/password> ::= current user's DB login name and password

<size> ::=

max size of backup volume, in K- or Mbytes. Default =K.

<Object description> ::= <Description codes>

<Description codes> ::=

```

-d
|-oref
|-oa<obj_name>
|-or<obj_name>
|-os<obj_name>
|-osr<obj_name>
|-ot<obj_name>
|-otwd<obj_name>
|-otr<obj_name>
|-op <obj_name>
|-ou<obj_name>
|-ov<obj_name>
|-oal l

```

```

<Incremental archive> ::= startinc | inc | stopinc | vi | wait
<Priority> ::= priority HIGH | priority NORMAL | priority LOW
<Tape backup> ::= dev<tname>[ crpart <pname>]
<tname> ::= name of tape for store/restore data.
<pname> ::= name of new tape partition
<Error handling> ::= { <-qq> | <-qc> | <-qx> } <event> [, ... ]
<qq> ::= terminate backup on event
<qc> ::= ignore event, continue backup
<qx> ::= wait for user input on event
<event> ::= { NV | DF | CB | OB | NB | ALL }

```

## Mode Options

cp	Set control points.
ef <file_name>	All commands is taken from specified file.
I	Display information about archive file.
R	Restore operation.
s	Backup operation.
t	Test the archive file.
tape	Operation involves tape.
script	lhb is to be run from a script, the name of which is the argument to the -ft option.
set	Set Russian or English as the language in which messages will be displayed. This mode command sets the language persistently, i.e., the language set will remain the default language when lhb is run until changed with the set argument. The default may be override for a single session by using the -le   -lr arguments.

### Remarks:

- 1) User names, passwords, and table names are case sensitive.
- 2) Commands and options are not case sensitive.

## Software Control Options

The software control keys allow you to set parameters for program functioning when backing up or restoring a DB. They also enable access control of lhb.

<u>Option</u>	<u>Argument</u>	<u>Description</u>
-c	<"comment">	Creates a user comment that is saved in the archive file.
-f	<archive_file>	Sets the path and name of the archive file. If not set, it defaults to the current directory and DB.lhb.
-fl	<script_file>	Sets the path and name of the log file. Can be used only for script mode.
-ft	<script_file>	Sets the path and name of the file which contains the backup script.
-g	<archive_password>	Sets a password for the archive. If the option is included without a password, the program will request password. If a DB archive is created setting a password, restoration will require the correct password. If the password is lost, restoration is impossible. <password> is not case sensitive.
-le, -lr		Display messages in English, -le or Russian, -lr. This setting will persist only for the current lhb session, after which the set mode option default is restored. These two options take no arguments.
-n	<server_name>	Sets the name of a remote Linter server. This option is required when backing up or restoring a remote DB.
-notesterc		Ignore BAD CRC
-n1, -p1, -u1		Backup the DB and replication server DB and at the same time DB and replication DB are not synchronized. These options are similar to -n, -p, -u.
-out	<msg_file>	File name to contain lhb messages for current session only. If the file already exists, its contents will be cleared first.
-out+	<msg_file>	File name to which lhb messages will be appended. If the file does not exist, it will be created.
-p	<path_to_restore>	Sets the path to the folder in which the DB will be restored. If option or argument is omitted, not specified, the DB will be restored in the current folder. If the DB is being restored to a folder already containing a DB, the program will ask the user whether to overwrite duplicate files.
-pg		Sets page-by-page mode of information display.
-synchronize	<file_name>	Synchronizes the DB and replication DB. The result batch file should be executed against the replication server DB
-u	<uname/password>	Provides user name and password to lhb for access verification.
-v	[<size>[ K   M ] ]	Sets multi-volume archive mode. The <size> parameter is an unsigned integer specifying the maximum size of an archive volume. The [ K   M ] argument specifies: Kilobytes

or Megabytes. The default is kilobytes. If size is undefined, it will be computed automatically. When creating automatic multi-volume archives, the program writes archive files to the storage media until there is no more free space, then requests the user to change the, or specify a different, medium. If the option is omitted, the program defaults to automatic volume creation.

-version Displays lhb number of version.

### Object Description Options

The object description keys are intended to provide the LHB program with information about DB objects, which are to be processed during backing up (restoring) of DB.

In the following option descriptions, the <obj\_name> argument specifies the object(s) the option is to use. This parameter may contain a single object name, e.g., the table name PERSON, or a series of objects in a comma-separated list.

In creating a list of objects pattern matching strings are supported with \*, meaning any number of symbols and the question mark ?, meaning a single symbol.

#### Examples

- s\* defines all objects whose name begins with S.
- \*doc\* defines all objects whose name contains the three letters doc.
- tab? defines all objects whose name differs only in the last symbol, i.e., tab1, tab2, tabs, etc.
- ot PERSON, AUTO, TST\_, BANK, \*bur\*

As you can see from the last example, pattern matching strings are treated the same as object name literals in that several may appear in the list, separated by commas.

<u>Option</u>	<u>Argument</u>	<u>Description</u>
-d		Forces lhb to back up or restore not only all other selected objects but also all objects that are dependent on the selected objects. For example, when this option is set, during back up of a table, all views and synonyms based on that table will also be backed up. However, objects connected according to the rules of link and reference integrity will not be automatically backed up. For such cases, use the -oref option. You can not use -d option with -oall option together.
-oa	<obj_name>	Sets privileges DB object(s). Each <obj_name> must be a table name for which access privileges are to be backed up or restored or a pattern matching string for such table names.
-oall		Selects all DB objects. This option is used for a complete DB backup: first, all DB objects are defined and then they are archived one by one. When performing a complete backup

<u>Option</u>	<u>Argument</u>	<u>Description</u>
		without this key, lhb creates the backup file-by-file. NB: when using -oall option, loss of data may occur if attempted during a multi-user session. This loss cannot occur when backing up file-by-file.
-only	<user_name>	Sets DB objects for particular user.
-op	<obj_name>	Sets procedures DB object(s). Each <obj_name> must be a procedure name or procedure name pattern matching string.
-or	<obj_name>	Sets roles DB object(s). Each <obj_name> must be a role name or a role name pattern matching string.
-oref		Sets the secondary keys DB object. This option is used to preserve DB link integrity. It forces lhb to save table data and all the links and references to and from the table in question.
-os	<obj_name>	Sets synonym DB object(s). Each <obj_name> must be a synonym name or a synonym name pattern matching string.
-osr	<obj_name>	Sets role assignment DB object. Each <obj_name...> must be a user name or a user name pattern matching string for users with assigned roles.
-ot	<obj_name>	Sets data table DB object(s). Each <obj_name> must be a table name or a table name pattern matching string. This option makes a complete table backup. The table may be restored later with all data intact.
-otr	<obj_name>	Sets triggers DB object(s). Each <obj_name> must be a trigger name or a trigger name pattern matching string
-otwd	<obj_name>	Backup table structure without data. <obj_name...> must be a table name or a table name pattern matching string.
-ou	<obj_name>	Sets user DB object(s). Each <obj_name> must be a user name or a user name pattern matching string.
-ov	<obj_name>	Sets views DB object(s). Each <obj_name> must be a view name or a view name pattern matching string.
-own		Force lhb to restore all foreign objects that were backed up with -takeforeign option. NB: when using -own for such objects as view, procedure, and triggers, loss data may occur because restoring procedure doesn't check if the objects related to foreign objects exist.
-takeforeign		Forces lhb to back up all foreign objects which is accessible.

## Incremental Archiving Options

Incremental archiving is used for backup when the DB must remain under active use. The options below are used to implement it.

<u>Option</u>	<u>Description</u>
-startinc	Begin incremental archiving.

-inc	Write collected changes.
-stopinc	Terminate incremental archiving.
-vi	Create new volume for increment
-wait	Wait for last updates

## Priority options Option

<u>Option</u>		<u>Description</u>
-priority	HIGH	Sets high priority, is used by default.
-priority	NORMAL	Sets normal priority.
-priority	LOW	Sets low priority.

## Control point options Option

<u>Option</u>	<u>Argument</u>	<u>Description</u>
-list		Show all control point.
-clear all		Clear all control points.
-clear	<point>	Clear the appointed control point.

## Tape Backup Options

The options in this category are used when working with archives stored on magnetic tape.

<u>Option</u>	<u>Argument</u>	<u>Description</u>
-dev	<tname> [,... ]	Path and name of tape device. Multiple tape devices will be written in the specified sequence.
-list		List files on archive tape <tname>.
-crpart	<pname>	Deletes all information on the tape and creates a new, blank partition named <pname>.

## Error Handling Options

These options determine lhb's reaction to error situations during the program execution. Multiple options are allowed to permit different handling of different error conditions.

The priorities of these options is in the sequence they are listed in the table below. If, for one type of event, several options are entered, the option with the highest priority is processed and the rest are ignored.

If the option lists no parameters, it is ignored.

The program recognizes the following error events, designated below by their argument values:

<u>Argument Value</u>	<u>Description</u>
ALL	All events listed below. CTL-BREAK received.
CB	CTL-BREAK received.
DF	File already exists.
NB	Version of Linter higher than expected.
NV	A new volume is needed.
OB	Wrong DB version during restore.

In the table below, the <event> argument is a single Argument Value from the table above or a comma-separated list of such values.

<u>Option</u>	<u>Argument</u>	<u>Description</u>
-qq	<event>	Terminate backup or restore when event encountered.
-qc	<event>	Ignore the event and continue backup or restore.
-qx	<event>	Wait for user input when event encountered.

## Launching lhb

To start lhb functions, the program is run from the operating system command line:

```
lhb {mode option ... } [ < option><argument> ... [ ,... ] ]
```

At least one mode option is required except when you want to see a list of lhb options and arguments, in which case, just enter lhb with no options or arguments.

## User Interface Language

If you want to set the language in which messages and screen displays will appear, you have the following choices:

<u>Option</u>	<u>Description</u>
set english	These mode options set the indicated language. When set, the language chosen becomes the default and remains such until the other language is set The following two options override the default for the current backup session only.
set russian	
-le	Select English for the current backup session only.
-lr	Select Russian for the current backup session only.

## Backup

Backing up the database is the process of saving selected or all DB objects in an archive file. During backup, the Linter kernel must be running. After backup, selected or all DB objects contained in the archive file may be restored. These objects may be restored into an already existing DB.

However, the type of backup used may restrict some of these choices; see the following two paragraphs.

You may do a complete backup either by using the `-oall` option or by setting all object description options, «Object Description Options», that are relevant to your DB. Considerations in making the choice include:

<u>Options</u>	<u>Description</u>
Using <code>oal</code>	Selecting all relevant Object
May suffer data loss if used multi-user environment	Will not suffer data loss if used in multi-user environment.
Every DB object defined, then written to the archive.	File-by-file backup
Can restore entire DB or restrict restoration to only some of the DB objects.	Must restore entire DB. Selective data object restoration is not possible. The DB being recovered is created anew in the directory specified by the software control option <code>-p</code> . See section “Software Control Options”.

Either complete or selective backup is possible while other DB users are active. Considerations regarding the archive created in a multi-user environment are described in «Scripting Operators, Comments, and Statements».

The following options may be used with selective backup:

```
-c      -ft      -n      -or      -osr      -v
-d      -g      -oa      -oref      -ot      -u
-f      -pg      -oall      -os      -ou      -y
```

The following options are used with complete backup:

```
-c      -g      -n      -f      -p      -u
-ft      -pg      -v
```

## Examples

1 Backing up the DB in file `base.lhb` with a comment to be included in the file:

```
lhb s -u SYSTEM/MANAGER -f base.lhb -c «Comment»
```

2 Backing up the DB automatically using the schedule contained in the script file `time.bsl` and setting a DB a password of `ABRIA`:

```
lhb script -u SYSTEM/MANAGER -g ABRIA -ft time.grf
```

## Restore

Restoring the DB is a process of creating objects identical to those contained in the archive file and if the backup was complete, creating a new DB.

During selective recovery, the Linter kernel must be running.

The process of complete restoration does not require Linter to be running.

Both complete and selective recoveries are possible when other users are logged on to the DB. However selective recovery in such a multi-user environment is not recommended since unexpected errors may occur.

Considerations regarding the archive created in a multi-user environment are described in «Scripting Operators, Comments, and Statements».

After the process of the restoration is completed Linter should be restated only for Linter version 5.9 and below.



Selective recovery in a multi-user environment is not recommended.

The following options may used during the restoration:

```
-d      -oall  -os      -otwd   -p      -f
-or     -osr   -ou     -pg     -oa     -oref
-ot     -ov    -u
```

## Examples

```
1 Complete      DB      recovery   from      the      archive   file      base.lhb
   contained in the current directory:
   lhb r -f base.lhb
```

```
2 Complete      DB      recovery   from      the      archive   file      base.lhb
   contained in C:\DB:
   lhb r -f base.lhb -p C:\DB
```

## Working with Archives

The following four sections describe considerations that are important in working with archives.

## Incremental Archiving

Incremental backup allows you to perform backup when it is impractical to shut down data base operations. It synchronizes continuing DB operations and the backup by logging all ongoing transactions and, when the backup is complete, writing those transactions into the archive.

Each of the three stages of an incremental backup are initiated by the command line options: startinc, inc, and stopinc.

Stage 1 e.g., lhb s -u SYSTEM/MANAGER -startinc -f base.lhb

-startinc lhb opens an open archive file. Linter expands the syslog file that logs transactions.

All transactions initiated after startinc are processed as usual to the DB. However, segments (extents) of the syslog file are not deleted when written to the DB.

When an extent is filled, Linter sets a synchronization mark and opens a new extent.

When the last extent has been filled, the first extent is opened and overwritten.



Simultaneously, lhb is writing to the open archive.

When the initial backup is complete, a synchronization point is set for it. The open

	archive must not be edited or deleted by a user.
Stage 2	e.g., <code>lhb s -u SYSTEM/MANAGER -inc -f base.lhb</code>
-inc	When this command is issued, the syslog is written to the open archive, one extent at a time, using the synchronization marks set for each extent and the open archive. The archive is no longer open; it is now a complete backup. The syslog extents are not deleted and continue to expand. At any time until you initiate stage 3, another stage 2 command may be issued.
Stage 3	e.g., <code>lhb s -u SYSTEM/MANAGER -stopinc -f base.lhb</code>
-stopinc	This command terminates the incremental backup and allows deletion of syslog extents that have been written to the DB.

The following options may be used during the incremental backup:

`-c -g -pg -f -n -v -ft -p -u`

-  When restoring the DB from an incremental archive, it is critical that the logged transaction changes to the open archive be written to the DB. Only then should the restoration procedure be started.
-  Because all delete transactions are retained as pending, the system log can grow to such gigantic proportions during periods of intense DB activity that the system may run out of disk space. It is important that the incremental backup not be left in an incomplete state for extended periods of time.

## Archiving on Tape

To create an archive on magnetic tape it is necessary to identify the tape drive and insure that it is cleared of any other data.

### Examples

```

1      Prepare tape
lhb tape -dev MT1: -crpart ARCHIVES

2      Identify tape to lhb
lhb type -dev MT1:

3      Backup DB to tape
lhb s -u SYSTEM/MANAGER -dev MT1: -f DBSTORE.lhb

4      View tape directory
lhb tape -dev MT1: -list

```

## Archive Status

This command allows the user to get general information about a tape archive.

The following options are available to archive status:

`-f -pg -g`

### Examples

```

The command
lhb l-f c:\linter\base.lhb
generates the following report.
ARCHIVE PASSWORD:          N/A
COMPRESSION:               STANDARD

```

```

USER:                SYSTEM
DB NAME:            TRIGG
DB VERSION:        5.4
ARCHIVE VERSION:    3.0.1
BACKUP TIME:        17.06.2006 14:39:23
CODING:            UNIX
COMMENT:

```

COMPRESSION shows that, when created, the archive was compressed to save storage space.

The value of ARCHIVE VERSION is the version number of the lhb program.

## Archive Testing

To test the integrity of the tape archive, the -t option is available.

During execution, checksums are calculated for all table objects and compared to the original checksums on tape.

When the test is complete, the same general information about the archive that shown in the status report, «Archive Status» is displayed. In addition, a table similar to the following shows the results for each table archived.

The following options are available to archive status

```
-f    -pg    -g
```

## Examples

The command  
lhb t -f c:\linter\base.lhb  
runs the test.

DB Object Type and Name	User	Result
(-)Table. AUTO	SYSTEM	100% OK
(-)Table... PERSON	SYSTEM	100% OK
(-)Table... \$\$\$COMMENTS	SYSTEM	100% OK
(-)Table... ALL_SEQUENCES	SYSTEM	100% OK
(-)Table... ALL_PRILTREE	SYSTEM	100% OK
(-)Table... M_COMPLECTATION	SYSTEM	100% OK
(-)Table... M_CUSTOMERS	SYSTEM	100% OK
(-)Table... M_TYPE_DOC	SYSTEM	100% OK
(-)Table... M_ED_IZM	SYSTEM	100% OK
(-)Table... TYPEINFO	SYSTEM	100% OK
(-)Table... PRIV_TYPES	SYSTEM	100% OK
(-)Table... \$\$\$PRCD	SYSTEM	100% OK
(-)Table... \$\$\$PROC	SYSTEM	100% OK
(-)Table... \$\$\$TRIG	SYSTEM	100% OK
(-)Table... \$\$\$DEVICE	SYSTEM	100% OK
(-)Table... \$\$\$STATION	SYSTEM	100% OK
(-)Table... \$\$\$RELATION	SYSTEM	100% OK
(-)Table... \$\$\$AUDIT	SYSTEM	100% OK
(-)Table... \$\$\$GROUP	SYSTEM	100% OK
(-)Table... \$\$\$LEVEL	SYSTEM	100% OK

The first column contains the DB object name. The minus, (-), indicates that the object was specified using an option. A plus, (+), would show that the object is dependant and was saved automatically.

The Result column shows the result of the test. The values in this column change from 0 to 100.

## Automating Backup

Backup is, or should be, one the most frequently performed administrative functions. Automating the process will not only save the administrator substantial time but will also provide a consistent backup procedure that insures the security and integrity of the DB.

lhb includes an integrated scripting language for creating scripts to perform all the necessary backup and restore functions.

The script contains the schedule for all backup operations and the sequencing of their execution by lhb DB backup, creating duplicate copies, transfer of archive and archive copies to various storage media, and removal of obsolete archives.

The following syntax description uses the Backus-Naur Form (BNF), with the following extensions:

<>	Angle brackets, encase a string which is a syntax element of the language
::=	he definition operator divides the element being defined on the operator's left from the definition on the operator's right
[ ]	Square brackets, indicate optional elements. They may be used or omitted.
	The or operator indicates that the element following it is one of more than one possible options.
...	The ellipsis shows that the preceding element may be repeated.
KEYWORDS	are in sans serif capital letters.
spaces	one or more, divide the syntax elements.

## Script Structure

The script will include some or all of the following sections:

- 1) Set time and date format;
- 2) Variables;
- 3) Execution schedule;
- 4) Special actions.

## Set Time and Date Format

The SET section of the script is optional. It is only included in the script if date or time type data will be used in a format different from the default. The user date and time format is set in the predefined variable TSFORMAT.

**Syntax**

```

SET: TIFORMAT = <date/time format>;
<date/time format>::=<format string>
<format string>::= <format element> [<delimiter><format element> ... ]
<delimiter>::=/|-|. |:|,| <space symbol> | «<symbol constant> »
<format element>::=
    <day of month> | <day of year> | <month> | <year> | <hour> | <minute>
<day of month>::= DD | dd
<day of year>::= DDD | ddd
<month>::= MM | mm | MONTH | Month | month | MON | Mon | mon | MONR
<year>::= Y [Y [Y [Y ] ] ] | y [y [y ] ] ]
<hour>::={HH | hh} [{MID | mid}] | {HH12 | hh12} [{MID | mid}] | {HH24 | hh24}
<minute>::= MI | mi
    
```

**Time and Date Format Description**

<u>Format String</u>	<u>Meaning</u>
DD   dd	Numeric day value; range 1 - 31.
DDD   ddd	Day of year; range 1 - 366.
MM   mm	Numeric month value; range 1 - 12.
MONTH   Month   month	Full name of month capitalized in the same way as the format string.
MONR	Roman numeral of month; range I - XII
Y   y	Year values; range 1-9
HH   hh   HH12   hh12	12 hour time format; range, 0 – 11.
HH24   hh24	24 hour time format; range, 0 – 23.
MID   mid	Include AM or PM in time; capitalized as per format string.
MI   mi	Include minutes in time; range is 0 – 59.
DEFAULT	DD/MM/YYYY HH24:MI

**Format examples**

Example	Format
16.02.1998:10:35	DD.MM.YYYY:HH:MI
16.02.98:10:35	dd.mm.yy:HH:MI
16-02-98 10:35	DD-mm-YY hh:mi
10 hrs 30 mins 16/02/1998	HH «hrs» MI «mins» DD/MM/YYYY
On December 31, 1999	«On» Month» «DD», «YYYY
10.35	hh.mi
Days until end of year: 264	«Days until end of year: « DDD
10.VII.2000	DD.MONR.YYYY
15/07/1999 07pm:30	DD/MM/YYYY hh12mid:mi
Last backup: MAR 23, 1998	«Last backup: « MON DD, YYYY

## Variables

The VARIABLES section of the script is also optional. It is used to define variables that differ from those predefined in the scripting language.

The language allows two types of variables symbols and numeric. The type of the variable is not explicitly declared. Type is determined by the initial value set for the variable.

### Syntax

```
VARIABLES: <variable definition> [ ; ...];
<variable definition> ::= <variable name> = <initial value>
<variable name> ::= <identifier>
<identifier> ::= alphanumeric string not longer than 18 symbols
<initial value> ::= »<symbolic constant>» | <numeric constant>
<numeric constant> ::= [ + | - ] <integer>
```

A symbolic constant can include \n, \t, \\, \» to define, respectively, new line, tab, the backslash itself, and quote.

### Examples

```
VARIABLES: FILE_DIR = «c:\linterSQL\db»;
VARIABLES: Name_File=»»; Count = 25;
```


## Scheduling Execution

The RIGHTS section of the script describes the dates and times at which the actions defined in the script are to be performed.

### Syntax

```
RIGHTS:
    { [<hourly operation> ... ]
      [<daily operation> ... ]
      | [<weekly operation> ... ]
      | [<monthly operation> ... ]
      | [<pre-set time operation> ... ] }
```

The sequence in which the operations are defined may be random.

 **For all operations in this section:** If, at any pre-set time more than one operation scheduled, the program will perform only one, ignoring any others. It is impossible to determine which operation will be executed. Less critical, but still a potential problem, are operations scheduled within a minute of each other. If, at the time set for operation B, lhb is performing <actions> defined by operation A, B's actions will be delayed until A's are complete. The script should be programmed so that there is at least a 1 minute interval between any two operations.

## Hourly

### Syntax

```
<hourly operation> ::= EVERYHOUR (min=<minutes>) <actions>
<minutes> ::= <minute of the hour for action>
<actions> ::= {<script functions>}
```

### Description

The user-defined <actions> will be performed on the set minute of every hour.

### Example

```

everyhour(min=59)
{
/* Backing up a highly dynamic table */
backup ( «s -u SYS/MAN -f c:\arc db -ot Orders»);
exception:
logprint(ctimestamp()+»:\n»);
logprint(«\tError=» + TOSTR(CERROR) );
logprint(«\tLinError=» + TOSTR(LINERROR) );
logprint(«\tSysError=» + TOSTR(SYSERROR) );
stop;
}

```

## Daily

### Syntax

<daily operation>::= **EVERYDAY** (time=<time>) <actions>  
 <time>::='< date in the HH24:MI format>  
 <actions>::={<script functions>}

### Description

The <actions> defined will be performed every day in at the time defined.

### Example

```

everyday(time='22:00')
{
/* Incremental backup every day */
backup(«s -inc -u SYSTEM/MANAGER -f «+ lastfilename );
}

```

## Weekly

### Syntax

<weekly operation>::= **EVERYWEEK** (day=<weekday>, time=<time>) <actions>  
 <weekday>::= Mon | Tue | Wed | Thu | Fri | Sat | Sun  
 <time>::='< date in the HH24:MI format>  
 <actions>::={<script functions>}

### Description

The <actions> defined will be performed every week on the day and time set.

### Example

```

everyweek(day=Mon,time='08:00' )
{
/* Creation of a new archive */
backup ( «s -startinc -u SYSTEM/MANAGER -f c:\lhb\»+
tostr(cdate( ) ) +«-« +
tostr(cmonth( ))+ «-« +
tostr(cyear2( ) ) +»\»» );
exception:
logprint(ctimestamp()+»:\n»);
logprint(«\tError=» + TOSTR(CERROR) );
logprint(«\tLinError=» + TOSTR(LINERROR) );
logprint(«\tSysError=» + TOSTR(SYSERROR) );
stop;
}

```

## Monthly

### Syntax

<monthly operation>::= **EVERYMONTH** (date=<day of month>, time=<time>) <actions>

```
<day of month>::=<date in the DD format> | LASTDATE
<time>::='< date in the HH24:MI format>'
<actions>::={<script functions>}
```

Lastdate is the last day of current month.

### Description

The < actions > defined will be performed every month on the day and time set.

### Examples

```
EVERYMONTH (date=4, time='03:00' )
EVERYMONTH (date=lastdate, time='23:59' )
```

## Specified Time

### Syntax

```
<at set time operation>::= DIRECT (timestamp=<date>) <actions>
<date>::=
    the date-time string in the format defined in the SET section or the default format.
<actions>::={<script functions>}
```

### Description

The <actions> will be performed once and at the time defined.

## Special Operations

The SPECIAL section describes actions that the lhb program must perform during execution or in error situations. This section is optional. If it is included, every operator must be mentioned only once.

### Syntax

```
SPECIAL:
    [<actions on start up>]
    [<actions on termination>]
    [<error handling actions>]
```

### Actions on Start Up

Usually, start up procedures will include such things checking available disk space, creating directories, testing previous lhb execution, and removal and re-creation of the archive.

These actions are performed when lhb is launched successfully.

If the script contains a RIGHTS section, see section “Scheduling Execution”, some of the usual start up procedures may be handled by those looping functions, hourly, daily, etc. The existence of a the RIGHTS section implies a perpetually running lhb. Real world necessity requires at least relatively, in the context of eternity, frequent restarts of lhb. Careful attention to the start up actions will relieve the you of having to rethink the start up procedures on the hopefully infrequent, in the context of this world occasions you need to launch lhb.

If the script does not contain a RIGHTS section the start up actions may include procedures usually handled in that section.

## Action on Termination

The <actions on termination> are performed when lhb terminates successfully; i.e., on CTL+C or stop commands. If lhb terminates abnormally, the < actions on termination are not performed.

This non-performance can be used to determine the way in which the prior run of lhb was terminated. If <action on startup> create a zero-length file and <action on termination deletes that file, the presence of the file on disk at startup shows the prior run was terminated abnormally.

## Error Handling Actions

The scripting language provides two structures for handling errors iferr and exception.

If included the SPECIAL section of the script, they provide error handling for errors generated elsewhere in the script for which no error handling is provided.

The following step describes lhb's error handling hierarchy:

1. If the current program block includes an iferr structure, the error is handled as defined in that iferr structure;
2. If the current program block does not include an iferr structure, the error is handled by the exception structure of the current program block;
3. If the current program block does not contain any error handling structure, lhb attempts to find one on a higher level of nesting. If an error handling structure is found on some nesting level, it is handled according to the criteria set in that structure;
4. If no error handling section is found inside other program blocks, the error is handled by the iferr structure in the SPECIAL section;
5. If the iferr structure is not found in the SPECIAL section, the error is ignored, a situation, which can lead to disaster.

## Actions on Startup

### Syntax

```
<actions on startup> ::= BEFORE <actions>
<actions> ::= {<script functions>}
```

### Description

The defined <actions> are performed at program launch.

## Actions on Termination

### Syntax

```
<actions on termination> ::= AFTER <actions>
<actions> ::= {<script functions>}
```

### Description

The <actions> are performed when the program terminates without an error or exception. A user command termination is not an exception.

## Error Handling Actions

### Syntax

```
<error handling actions> ::= IFERR <actions>
<actions> ::= {<script functions>}
```

### Description

The <actions> are performed when the program encounters an error during execution.

## Scripting Operators, Comments, and Statements

### Operators

The following operators may be used with strings:

<u>Operator</u>	<u>Purpose</u>	<u>Result or Type</u>
+	Concatenation	String
=	Assignment	String
==	Equality comparison	0 = False; 1 = True
!=	Inequality comparison	0 = False; 1 = True

The following numeric operators are available:

<u>Operator</u>	<u>Purpose</u>	<u>Notes</u>
+	Addition	Binary operation
-	Subtraction	Binary operation
-	Sign change	Unary operation
=	Assignment	
==	Equality comparison	0 = False; 1 = True
!=	Inequality comparison	0 = False; 1 = True
>	Greater than	0 = False; 1 = True
<	Less than	0 = False; 1 = True
>=	Greater than or equal	0 = False; 1 = True
<=	Less than or equal	0 = False; 1 = True
AND	Logical AND	0 = False; 1 = True
OR	Logical OR	0 = False; 1 = True
NOT	Logical negation	0 = False; 1 = True

### Comments

A comment may begin at any position on a line between syntax elements of the source text.

It cannot contain a new line and must be terminated with a new line.

`/*` precedes a comment. It is terminated with `*/`

## Scripting Statements

### Loops

#### Syntax

```
WHILE ( <condition> ) {<operator(s)>};
<condition>::= an expression with a numeric type result.
```

#### Description

<Operator(s)> are evaluated repeatedly until the <condition> result is equal to zero.

### Example

```
while(I)
{ Description:
If the <selection condition> is not equal to zero then <script function1 > is
evaluated, otherwise, when else is present,
<script function2> is evaluated.
Examples:
I=I-1;
/* incremental archive */
backup ( «s -inc -u « + myname + «/» + mypass +
-f « + «\»» + lastfilename+ «\»»);
if(CERROR==0) logprint(ctimestamp()+»:»:»+ »>>+lastfilename);
}
```

### Conditional

#### Syntax

```
IF (<selection condition> ) { <script function1 > }
[ ELSE
{ <script function2> } ];
<selection condition>::= expression with a numeric type result.
```

#### Examples

```
/* Ex.1: If the user did not specify username and password */
if ( name == «» )
{
MYNAME = GETSTR( «Username: « );
MYPASS = GETPASS( «Password: « );
} else
{
MYNAME = NAME;
MYPASS = PASSWORD;
}

/* Ex.2: create directory for archive */ if ( not exist ( FILES_DIR ) )
crdir ( FILES_DIR );
```

### Backup

#### Syntax

```
BACKUP ( «<command line>» ) [ iferr { <error handling> } ];
<command line>::=<character string>
```

#### Description

This statement launches lhb with parameters specified in the <command line>.

The script can contain only one backup statement.

The iferr structure will be evaluated only on error. If the structure is not included the error will be evaluated according to the error handling hierarchy. See section “Special Operations”.

#### Example

```
backup ( «s -startinc -u « + myname + «/» +
mypass + « -f « + «\»» +
FILES_DIR + «\» + tostr(cdate( ) ) + «-« + tostr(cmonth( ) ) +
«-« + tostr(cyear2( ) ) + «\»» );
```

### Copy File

#### Syntax

```
COPY (<source file> , <target file> ) [ iferr {<script functions>} ];
```

<source file>::=<character string>  
<target file>::=<character string>

### Description

This statement copies <source file> to <target file>. <character string> includes any required path elements and may not include wild cards such as the asterisk, \*. The file being copied is not altered in any way.

### Example

```
copy («251298.lhb», »c:\arc_db\251298.lhb»);
```

## Move File

### Syntax

```
MOVE (<source file> , <destination> ) [ iferr {<script functions>} ];  
<source file>::=<character string>  
<destination>::=<character string>
```

### Description

This statement moves <source file> to <destination>. <character string> includes any required path elements and may not include wild cards such as the asterisk, \*.

### Example

```
move («251200.lhb», »/xdb/arc_db/251200.lhb»)
```

## Rename File

### Syntax

```
RENAME (<original file>, <renamed file> ) [ iferr {<script functions>} ];  
<original file>::=<character string>  
<renamed file>::=<character string>
```

### Description

This statement assigns a new name to a file. The location of the file is not altered.

### Example

```
rename («c:\arc_db\251200.lhb», »251200.arc»);
```

## Delete File

### Syntax

```
DELETE ( <target file> ) [ iferr {<script functions>} ];  
<target file>::=<character string>
```

### Description

This statement deletes the target file. The file name cannot include wild cards such as the asterisk, \*.

### Example

```
delete («../old/241200.lhb»);
```

## Create Directory

### Syntax

```
CRDIR ( <dir name> ) [ iferr {<script functions>} ];  
<dir name>::= <character string>
```

### Description

This statement creates a directory.

## Launch Program

### Syntax

```
RUN ( <program name> ) [ iferr {<script functions>} ];  
<program name>::= <character string>
```

### Description

This statement launches the named program. lhb goes into idle mode until the named program has terminated. <Program name> includes any required path elements. If the path to the program is not specified, the program will be searched for in the current directory. The result of program execution is returned in the predefined variable CERROR.

## Execute Program

### Syntax

```
EXEC ( <program name> ) [ iferr {<script functions>} ];  
<program name>::=<character string>
```

### Description

This statement launches the specified program and lhb continues to run. <program name> specifies the program name and location. If the path to the program is not specified the program will be searched for in the current directory. The result of program execution is not returned.

## Exit lhb

### Syntax

```
STOP;
```

### Description

The stop statement terminates lhb execution.

## Break

### Syntax

```
BREAK;
```

### Description

Break terminates execution of an if or while execution.

## Exception Handling

### Syntax

```
EXCEPTION;
```

**Description**

This statement marks the beginning of the error handling section in the current nesting level. See section «Special Operations».

**Error Handling****Syntax**

```
ERROR;
```

**Description**

This statement accesses the exception section or the iferr structure. See section “Special Operations”.

**Predefined Variables and Constants**

<u>Variable</u>	<u>Value</u>
LASTFILENAME	The path and name specification of the last archive file accessed
LASTDATE	Date of last backup
CERROR	LHB termination code or the termination code of the program executed using the RUN operator. This code is checked before the call to iferr is made
LINERROR	Termination code returned by DBMS Linter
SYSERROR	Termination code returned by the O/S
NAME	Username from the command line
PASSWORD	Password from the command line
FILENAME	Filename from the command line
DATEFORMAT	The date-to-string conversion format

**Predefined Functions**

<u>Function</u>	<u>Description and Return Value</u>
CTIMESTAMP ( )	Returns current date in TSFORMAT format.
CYEAR(2)	Returns 2 digit current year.
CYEAR(4)	Returns 4 digit current year.
CMONTH( )	Returns numeric current month in range of 1 - 12.
CDAY( )	Returns numeric current day of week. Mon.= 1.
CWEEKDAY( )	Returns first 3 characters of current weekday name.
CDATE( )	Returns numeric current day of month in range of 1 – 31.
CHOUR( )	Returns numeric current hour.
CMIN( )	Returns numeric current minute.
EXIST(<file>)	0 = file does not exist; 1 = file does exist.
PRINT(“<text>”)	Prints <text> to console as a message. Console is monitor on which lhb is running. Always returns 1.

LOGPRINT (“<text>”)	Prints <text> to backup log. If log does not exist, function is ignored. Always returns 1.
TOSTR (<numeric expr.>)	Returns string converted from <numeric expr.>.
GETSTR (“<prompt>”)	Returns console input as user’s response to <prompt>. Prompt cannot exceed 200 symbols. Example: GETSTR(«lhb>Enter file name of archive: «)
GETPASS («<prompt>»)	Returns password entered at console in response to <prompt>. As password entered, asterisks, *, echoed. Maximum prompt is 200 symbols. Example: GETPASS(“lhb>Password?: “)

## Sample Script

```

/* ----- */
/* - Sample script - */
/* ----- */
Variables:
NUM = 1;
Rights:
Everyday (time = '12:00')
{
/* New archive created on Sundays */
if (CWEEKDAY( ) == «Sun»)
{
move (FILENAME + TOSTR(NUM) + «.lhb» , «c:\arc» );
NUM = NUM + 1;
backup («s -u « + NAME + «/» + PASSWORD +
« -f « + FILENAME + TOSTR(NUM) +
«.lhb -startinc»);
}
/* Other days use incremental archive */
else
{
backup («s -u « + NAME + «/» + PASSWORD +
« -f « + FILENAME + TOSTR(NUM) + «.lhb -inc» );
} /* end of 'if */
Exception: /* Error handling */
print ( «Error=» + TOSTR(CERROR) +
« , LinError=» + TOSTR(LINERROR) +
« , SysError=» + TOSTR(SYSEERROR) );
stop;
}
Special:
before /* do immediately after launch */
{
print ( «Start backup system» );
backup ( «s -u « + NAME + «/» + PASSWORD +
« -f « + FILENAME + TOSTR(NUM) + «.lhb -startinc» );
}
after /* after 'stop' or Ctrl-C */
{
print ( «Stop backup system» );
if ( ERROR != 0 )
logprint ( «Error present: « + TOSTR(ERROR) ); }
iferr /* global */ {
print ( «Error=» + TOSTR(CERROR) +
« , LinError=» + TOSTR(LINERROR) +
« , SysError=» + TOSTR(SYSEERROR) );
stop;
}

```

## Backup of a Working DB

Linter provides a secure method of backing up a DB that cannot be shut down, incremental backup, see section “Incremental Archiving”, to accommodate the backup.

Other backup procedures are designed to perform backup when the DB is not in use.

You should look at each of the following circumstances to determine if an alternative to incremental backup is reasonable in light of the risks.

During selective backup, lhb does not backup an already archived object which has been changed by a users during the backup. That is, if after lhb has backed up an object, a user modifies it, that modified data will not be saved in the archive. It will not be included in a subsequent recovery.

Assume that, during selective backup, the COMMIT operation for a transaction is executed when the backup manager is running, but after it has backed up the transaction objects and the system log. In this case, the changes made by the transaction will be lost when recovering data from the archive.

During complete backup of an active DB the data is archived in its entirety. If, after the backup is complete, there are some unfinished transactions, those transactions will not be available in a subsequent recovery.

## lhb Messages

### Execution Report

Structure of the diagnostic message:

<u>Line N°.</u>	<u>Content</u>
1	LINTER HOT BACKUP PROGRAM — Ver 3.1.0 build 326 — Relex, Ltd. 1996-2006
2	<execution report line>
...	
2	<execution report line>
3	lhb error (<lhb exit code>) <message text>
4	Linter error (<exit code>) <message text>
5	System error (<exit code>) <message text>

### Description

<u>Line N°.</u>	<u>Content</u>
Line1	lhb ID information.
Lines2	lhb ID information.
Line3	exit code and message text of the last operation that failed.

Line4 Linter exit code and message text.  
 Line5 operation system exit code and message text.

For lhb exit codes and messages, see document “Termination Codes”.

Linter exit codes and messages are located in Appendix A

OS system exit codes and messages will be located in the OS documentation.

## Common Messages, Causes, and Suggestions

The lhb backup manager returns the following messages:

<u>Message</u>	<u>Cause</u>	<u>Suggestions</u>
Connection error	1 The database local or remote which is to be backed up is inactive. 2 The remote Linter server name has been entered incorrectly. 3 A network error, e.g., a network driver malfunction.	1 Active the DB in question and retry. 2 Check the server name. 3 Contact the network administrator.
Wrong password	The password entered by the user and the password kept in the archive do not match.	Enter the correct password. NB: The password is case sensitive.
Cannot open file <filename>	During recovery or testing or when using other commands, a nonexistent filename has been specified or the user does not have access rights to the file.	Specify the correct path and/or filename or grant the user necessary access rights.
Incorrect archive <filename>	A file has been specified that is not a DB archive.	Enter the correct file specification.
Version of <filename> file obsolete	A file created by an earlier version of the backup manager has been specified.	Use the version of lhb that corresponds the version used to create the file.
LHB version obsolete	The archive file was created using a newer version of lhb than your current version.	Use the version of lhb that corresponds to the version used to create the archive.
Incorrect password	Incorrect password entered.	Enter the correct password. If the password is lost then the archive is lost forever.
Begin new volume?	The backup command included the -qq NV option.	Answer «Yes» if a new volume is needed or «No» if not.
Illegal date format <date>, must be 'DD.MM.[YY]YY.H H:MI'	In the script file, the date assignment is in the wrong format.	Correct the script format error.
Control sum compare error	The archive's integrity has been compromised.	Recovery of a bad archive is impossible. If a duplicate exists use the duplicate.

<u>Message</u>	<u>Cause</u>	<u>Suggestions</u>
File already exists. Delete?	1 An already existing filename has been specified in the backup command.  2 An already existing DB was specified in the recovery command.	If the answer is «Yes» the file will be replaced, if the answer is «No», lhb will exit.

## Termination Codes

<u>Message</u>	<u>Code</u>	<u>Cause</u>
ERR_TAPE_FINDBEG	30100	Could not find the beginning of tape
ERR_TAPE_READ	30101	Read from tape error
ERR_TAPE_OPEN	30102	Tape open error
ERR_TAPE_CLOSE	30103	Tape close error
ERR_TAPE_WRITE	30104	Write to tape error
ERR_LHB_OPENFILE	30105	Open file error
ERR_LHB_BREAK	30106	* Program terminated on CTRL+C
ERR_LHB_CRC	30107	Control sum error
ERR_LHB_READFILE	30108	Read file error
ERR_LHB_WRITEFILE	30109	Write file error
ERR_LHB_DELETEFILE	30110	Delete file error
ERR_LHB_DBOPEN	30111	Connect to DB error
ERR_LHB_STARTLHB	30112	Backup start error
ERR_LHB_STOPINC	30113	Could not finish incremental backup
ERR_LHB_READOLDBACKUP	30114	Could not find incremental archive information
ERR_LHB_WRITEOLDLHB	30115	Could not write to incremental archive
ERR_LHB_SETLOGNOTREUSE	30116	Log reuse ban error
ERR_LHB_CLRLOGNOTREUSE	30117	Log reuse allow error
ERR_LHB_STOREUSER	30118	Store user information error
ERR_LHB_STOREVIEW	30119	Store view error
ERR_LHB_STORESYN	30120	Store synonym error
ERR_LHB_STOREROLE	30121	Store role error
ERR_LHB_STORESETROLE	30122	Store role assignment error
ERR_LHB_STOREACCESS	30123	Store access rights error
ERR_LHB_STOREFOR	30124	Store secondary keys error

<u>Message</u>	<u>Code</u>	<u>Cause</u>
ERR_LHB_RESTOREACCESS	30125	Restore access rights error
ERR_LHB_RESTOREUSER	30126	Restore DB users error
ERR_LHB_RESTOREROLE	30127	Restore role error
ERR_LHB_RESTORESETROLE	30128	Restore role assignment error
ERR_LHB_RESTORESYN	30129	Restore synonym error
ERR_LHB_RESTOREFOR	30130	Restore secondary keys error
ERR_LHB_RESTOREVIEW	30131	Restore view error
ERR_TAPE_NOTLHBPART	30133	LHB-partition on tape search error
ERR_TAPE_FILEEXIST	30134	File already exists on tape
ERR_TAPE_INVDATA	30135	Data read from tape error
ERR_TAPE_NOTENDOFDATA	30136	Could not find tape end
ERR_LHB_OLDLINTER	30137	DBMS Linter version obsolete
ERR_LHB_NEWLINTER	30138	DBMS Linter version too new
ERR_LHB_RENAMEFILE	30139	Could not rename file
ERR_LHB_MKDIR	30140	Could not create directory (folder)
ERR_LHB_RUN	30141	Program launch error
ERR_LHB_DECOMPRESS	30142	Unknown compression method
ERR_LHB_NOTEQLINTER	30143	DBMS Linter version does not correspond to the archive version
ERR_LHB_INVCRYPTO	30144	Incorrectly coded data
ERR_LHB_INVFILE	30145	Bad archive file
ERR_LHB_NOMEM	30146	Not enough memory
ERR_LHB_TAPEFREE	30147	No data on tape
ERR_TAPE_LOCK	30148	Block tape error
ERR_TAPE_UNLOCK	30149	Unblock tape error
ERR_LHB_OPEN_INC	30150	Could not find incremental archive file
ERRAPE_BADEND	30151	Could not find tape end
ERR_TAPE_INCNOTEND	30152	Incremental archive file not finished
ERR_TAPE_SEEK	30153	Tape seek error
ERR_TAPE_SETBLSIZ	30154	Tape block size setting error
ERR_TAPE_SETFM	30155	File marker write error
ERR_TAPE_FINDFM	30156	File marker find error
ERR_LHB_NOTSUPPORT	30157	The program does not support the given operation

\* Returned in operator After of section Special.