jBASE Utilities

Contents

Documentation Conventions

This manual uses the following conventions:

Convention	Usage
BOLD	In syntax, bold indicates commands, function names, and options. In text, bold indicates keys to press, function names, menu selections, and MS-DOS commands.
UPPERCASE	In syntax, uppercase indicates JBASE commands, keywords, and options; BASIC statements and functions; and SQL statements and keywords. In text, uppercase also indicates JBASE identifiers such as filenames, account names, schema names, and Windows NT filenames and pathnames.
UPPERCASE *Italic*	In syntax, italic indicates information that you supply. In text, italic also indicates UNIX commands and options, filenames, and pathnames.
Courier	Courier indicates examples of source code and system output.
Courier Bold	**Courier Bold** In examples, courier bold indicates characters that the user types or keys (for example, <Return>).
[]	Brackets enclose optional items. Do not type the brackets unless indicated.
{}	Braces enclose nonoptional items from which you must select at least one. Do not type the braces.
ItemA | itemB	A vertical bar separating items indicates that you can choose only one item. Do not type the vertical bar.
. . .	Three periods indicate that more of the same type of item can optionally follow.
⇒	A right arrow between menu options indicates you should choose each option in sequence. For example, “Choose **File** ⇒ **Exit**” means you should choose **File** from the menu bar, and then choose **Exit** from the File pull-

Syntax definitions and examples are indented for ease in reading.

All punctuation marks included in the syntax—for example, commas, parentheses, or quotation marks—are required unless otherwise indicated.

Syntax lines that do not fit on one line in this manual are continued on subsequent lines. The continuation lines are indented. When entering syntax, type the entire syntax entry, including the continuation lines, on the same input line.

GENERAL UTILITIES

Customizing the output of commands

JBASE contains a variety of utilities of which the use of some provides an alternative interface to UNIX commands while others enable or disable certain jBASE features.
It is possible for a customer to modify the output characteristics of the WHERE, LISTU and SHOW-ITEM-LOCKS utility programs by creating a subroutine called by these programs, which are accessible to all users who will call the programs. The name of this fixed subroutine is: JBCUserCustomiseDisplay

It is passed to a single parameter, which is both the input and the output parameters and is called as a common subroutine for WHERE, LISTU and SHOW-ITEM-LOCKS. It is called once at the start of each program to present the heading information, and once again for each line of output data to the screen (or printer). The subroutine can choose to modify the headers and output data as required, and can do so differently for each command, or add it's own options. For example: a new command line option such as (U) could be added on the command line to specify that it should display the user location details.

There are two ways to call a subroutine.

1. Header Specification

It presents the following information to the caller in the single parameter:

attribute 1	One of "WHERE", "LISTU" or "SHOW-ITEM-LOCKS"
attribute 2	Always 1 to show it is presenting the headers.
attribute 3	A multi-value list of the heading text, one value per column.
attribute 4	A multi-value list corresponding to attribute 3, which has numeric definitions to describe the column. Contains these definitions in the supplied JBC.h file. This makes it more portable to find, for example, in which column it displays the port number. Use this code, to find in which column number it presents the port number. INCLUDE JBC.h LOCATE CUSTOM_TYPE_PORT_NUMBER in parameter<4> SETTING ....
attribute 5	This is the width of each heading column. This value does not include the delimiter added by the jBASE program.

Body Specification

It calls the subroutine once for each line of body data that is output by the jBASE program. When completed it presents the following information to the caller in the single parameter

attribute 1	One of "WHERE" , "LISTU" or "SHOW-ITEM-LOCKS". Do not amend this attribute
attribute 2	Always "2" to show it is presenting the data. Do not amend this attribute.
attribute 3	A multi-value list of the data it wishes to display. Each value corresponds to the layout in the heading as presented in the first call shown above. For example, if it presented the account name in the 3rd column of the output of WHERE, then the account name will be in the 3rd multi-value of this attribute 3, and the first call, which defines the headings, will have multi-value 3 in attributes 3,4 and 5 defining the account name. If you want to remove a complete row of data, then in attribute 3 return a null string.

NOTES

When finding locating in which column it holds a particular piece of information, **REMEMBER**: if you insert or delete fields, it could invalidate that field number. The order of finding the information is important. For example, you might use LOCATE (as in the examples) to locate the column in which the Account name appears during the WHERE command, usually by default is column 3. If you later decide to delete the Port number, in column 1, then it displays the Account name in column 2 instead of column 3.

The WHERE command, has the (V) option, which means it displays multiple lines of information for each jBASE active program. When this happens you probably want to mimic the layout of the WHERE command i.e. it only displays the columns such as Port, Device, Accounts on the first row of data for each jBASE process

The 'Command' column on the WHERE command and the 'FILENAME' and 'RECORD KEY' columns for the SHOW-ITEM-LOCKS commands vary in width and are initially created so the command fits the entire width of the available screen. If you modify the columns to display, the jBASE commands WHERE and SHOW-ITEM-LOCKS will re-calculate their variable length fields to fits onto the screen width. The LISTU command has no such fields; therefore, the programmer must ensure all modified fields will fit onto the screen width.

EXAMPLE 1

Remove the “Device” column from the output of the WHERE command. The output of a WHERE command that is:

Port	Device	Account	PID	Command
10	Ttyp0	Jbase32	8838	Jsh
		8840	ED	TASKLIST
			BP	
12	Ttyp2	JBASE32	11328	WHERE
14	Ttyp1	JBASE32bl	11327	ADD.CUSTO
		d		MER

Is changed to:

Port	Account	PID	Command
10	JBASE32	8838	Jsh
	8840	ED BP	TASKLIST
*12	JBASE32	11328	WHERE
14	JBASEbld	11327	ADD.CUSTOMER

This is the required subroutine.

```
    SUBROUTINE JBCUserCustomiseDisplay(rec)

 Customize so that for the WHERE command only, we will not display
the DEVICE column.

 In order to keep the functionality quicker, we will use named
commons
* so we aren’t permanently opening and closing files.

INCLUDE JBC.h
COMMON /JBCUserCustomiseDisplay/ device.column
IF rec<1> EQ “WHERE” THEN ; Limit ourselves to the WHERE command
   IF rec<2> EQ “1” THEN

 This is the call that defines the heading statements.
* Look to see where the device name is stored.
*
      LOCATE CUSTOM_TYPE_DEVICE_NAME IN rec<4> SETTING device.column
THEN
*

* Delete the three values that define, for the device name, the
heading
* text, the heading identifier and the heading width.
*
            DEL rec<3,device.column>
           DEL rec<4,device.column>
           DEL rec<5,device.column>
       END ELSE
           device.column = zero
```

```
        END
    END ELSE
*
* this is a line of data. Remove the data for the device name
* as we did for the heading statement.
*
            IF device.column THEN
           DEL rec<3,device.column>
       END
  END
END

RETURN
```

EXAMPLE 2

In this example we:

Remove the device name column as we did in the first example.

Insert a new column of our own which we will call ‘Location’; data we create ourselves will fill this column. In this example, it is simply a cross-reference between the port number and a record keyed on the port number.

The account name field will change to a width of 16. This will allow us extra room to add a (S) to the account name if we find the user is in sales.

Therefore, the output of WHERE will change from:

Port	Device	Account	PID	Command
10	Ttyp0	Progerm	8838	Jsh
	8840	ED	BP	Newshell
*12	Ttyp2	JBASE32	11328	WHERE
14	Ttyp1	Dorisk	11327	ADD.CUSTOMER

to:

Port	Account	Location	PID	Command
10	Rogerm(S)	Sales Station	128838	Jsh
	8840	ED	BP	Newshell
*12	JBASE32	Developer Desk	11328	WHERE
14	Dorisk	Marketing SW	11327	ADD.CUSTOMER

Here is the required code.

```
SUBROUTINE JBCUserCustomiseDisplay (rec)

 Customize ourselves purely for the WHERE command.
*     The output for WHERE is normally like this:
* Port     Device     Account     PID     Command
```

```
* We will modify the output in the following manners
*    (a) Delete the Device column
*    (b) Insert a new "Location" field, preferably after the
* Account field
*    (c) Update the Account column to be 16 characters and
* append (S) for sales
* This means the heading now looks like:
* Port     Account          Location         PID      Command

 In order to keep the functionality quicker, we will use named
commons
* so we aren't permanently opening and closing files.
*
INCLUDE JBC.h
COMMON /JBCUserCustomiseDisplay/ FILEVAR, delvalue
COMMON /JBCUserCustomiseDisplay/ insvalue, accvalue
COMMON /JBCUserCustomiseDisplay/ okayflag, portnumber
*
* This function is called by many commands other than
* the WHERE command,

command = rec<1>; Extract the name of the command type
= rec<2>        ;* Extract the type of call.
BEGIN CASE
 CASE command EQ "LISTU"      ;* Ignore the LISTU command (for
clarity)
 CASE command EQ "SHOW-ITEM-LOCKS" ;* Ditto
 CASE command EQ "WHERE"
* It is only the WHERE command we are interested in.
* The other CASE statements added for clarity of code only

 See if this is a heading statement or line of details
*
IF type EQ "1" THEN
*
* This is the heading definition. The rest of the 'rec' is made up
as follows
* attribute 3: Multi-value list of heading text
* attribute 4: Multi-value list of heading definitions
* attribute 5: Multi-value list of widths of each column

 Open the LOCATION file for our use.
*
      OPEN "LOCATION" TO FILEVAR ELSE
            okayflag = zero
            RETURN
      END
      okayflag = one
*
* Find out where we can extract the port number from.
*
 LOCATE CUSTOM_TYPE_PORT_NUMBER IN rec<4> SETTING portnumber ELSE
      okayflag = 0
      rec = recsave
      RETURN
 END
*
* (a) Find the Device Name to delete

 recsave = rec
 LOCATE CUSTOM_TYPE_DEVICE_NAME IN rec<4> SETTING delvalue THEN
      DEL rec<3,delvalue>     ; Delete the heading text
                DEL rec<4,delvalue>     ;* Delete the definition
                DEL rec<5,delvalue>     ;* Delete the width
           END ELSE
                delvalue = 0  ;* Cannot find the heading
           END
```

```
 (b) We will add the 'Location' value AFTER the account name

 LOCATE CUSTOM_TYPE_ACCOUNT_NAME IN rec<4> SETTING insvalue THEN
              insvalue++    ; This is the value we insert BEFORE
           END ELSE
              insvalue = 2  ;* Default to becoming the second
column
           END
           INS "Location" BEFORE rec<3,insvalue> ;* Insert the text
           INS "99" BEFORE rec<4,insvalue>       ;* Insert a dummy
value for the heading definition
           INS "20" BEFORE rec<5,insvalue>       ;* Insert the width
to use

 (c) Find the column of the Account definition
*
           LOCATE CUSTOM_TYPE_ACCOUNT_NAME IN rec<4> SETTING
accvalue ELSE
              accvalue = 0
           END*
       END ELSE

 This is a line of data in attribute 3 Each column is
* multi-valued We will amend this line of data according
* to the data we extracted earlier.

 Make sure the OPEN went okay
*
      IF NOT(okayflag) THEN
      RETURN
END
*

* Extract the port number before we do anything else.
*
port = rec<3,portnumber>
*
* (a) Delete the Device Name
*
IF delvalue THEN
DEL rec<3,delvalue>
           END
*
* (b) Insert the LOCATION information.
*
 IF port NE "" THEN
 READ location FROM FILEVAR, port ELSE
location = "UNKNOWN"
END
      END ELSE
              location = ""
           END

           IF insvalue THEN
              INS location<1> BEFORE rec<3,insvalue>
           END
*
* (c) Amend the account name to have (S) appended if a sales account

           accname = rec<3,accvalue>   ; The account name or ""
           IF NOT(LEN(accname)) THEN
              location = ""
           END
           IF location<2> = "SALES" AND LEN(accname) THEN
              rec<3,accvalue> = accname :" (S)"
           END
       END
```

```
END CASE
RETURN
```

ADDD

The jBASE ADDD command adds together two decimal numbers.

COMMAND SYNTAX

```
ADDD Num Num
```

SYNTAX ELEMENTS

Num decimal number

EXAMPLE

ADDD 89 10

99

ADDX

The jBASE ADDX command adds together two hexadecimal numbers.

COMMAND SYNTAX

```
ADDX HexNum1 HexNum2
```

SYNTAX ELEMENTS

Num hexadecimal number

EXAMPLE

ADDX 59 0A

63

AUTOLOGOUT

The AUTOLOGOUT command enables a session to log off automatically should there be no keyboard activity for a specified time.

COMMAND SYNTAX

```
AUTOLOGOUT {n {program {program Args}}} {©}
```

SYNTAX ELEMENTS

n is the number of minutes of keyboard inactivity required before logging the session off.
program is the name of a program to run before logging the session off.
program specifies args for the program.
The (C option displays the name of the program to be executed.

NOTES

The definition of inactivity is as a program waiting for an INPUT or IN statement.
If no program is specified when it issues the AUTOLOGOUT command, it executes a program called ON.EXIT before logging off the session. The user must define the functionality of ON.EXIT. To reset the auto logout, specify zero for the number of minutes, or disconnect and reconnect from jBASE.

EXAMPLES

jsh -->AUTOLOGOUT 10

Automatic logout is set for 10 minutes.
jsh -->AUTOLOGOUT 15 OFF (C

Automatic logout is set for 15 minutes
.
Command to execute when invoked = ‘OFF’

BREAK-KEY-ENABLE

The jBASE BREAK-KEY-ENABLE command sets the jBASE break key enabled flag. This enables the break key, when executing jBASE programs.

COMMAND SYNTAX

```
BREAK-KEY-ENABLE
```

SYNTAX ELEMENTS

None

EXAMPLE

```
BREAK-KEY-ENABLE
```

BREAK-KEY-OFF

The jBASE BREAK-KEY-OFF command resets the jBASE break key enabled flag. This disables the break key, when executing jBASE programs.

COMMAND SYNTAX

```
BREAK-KEY-OFF
```

SYNTAX ELEMENTS

None

EXAMPLE

BREAK-KEY-OFF

BREAK-KEY-ON

The jBASE BREAK-KEY-ON command sets the jBASE break key enabled flag. This enables the break key, when executing jBASE programs.

COMMAND SYNTAX

```
BREAK-KEY-ON
```

SYNTAX ELEMENTS

None

EXAMPLE

```
BREAK-KEY-ON
```

CLEAR-ITEM-LOCKS

Use the CLEAR-ITEM-LOCKS command to clear a specific lock or all locks taken against a specific file. Use this command with jBASE hashed files only.

COMMAND SYNTAX

CLEAR-ITEM-LOCKS filename

CLEAR-ITEM-LOCKS filename, itemname

NOTES

Only a user with root privileges can issue this command

.

This is a UNIX/LINUX only command

.

Refer to the SHOW-ITEM-LOCKS command

.

If JEDI_NOSHMEM is set (i.e. JEDI_NOSHMEM=1) the CLEAR-ITEM-LOCKS will not function (?)

COMO

The COMO utility provides a means of recording all input and output from the terminal to a specified record in a directory.
On UNIX platforms the directory is called &COMO&; called COMO on Windows.

COMMAND SYNTAX

COMO ON SessionId

Turn recording on to SessionId
COMO OFF

Turn recording off
COMO DELETE SessionId

Delete recording named SessionId.

NOTES

Use the UNIX command typescript for this functionality if COMO is NOT part of your legacy application.
E.g; script - Start recording

command execution

Ctrl-D - Stop recording.
You can find Recorded information in the “typescript” file.

CONTROL-CHARS

Use the CONTROL-CHARS command to control whether or not the jBASIC INPUT statement accepts characters outside the printable range (decimal 32 through 126).

COMMAND SYNTAX

```
CONTROL-CHARS {ON | OFF}
```

SYNTAX ELEMENTS

CONTROL-CHARS ON	Disallows the input of control characters.
CONTROL-CHARS OFF	Allows the input of control characters.
CONTROL-CHARS	Reports the current state.

NOTES

The default behavior is to allow the input of control characters. The jBASIC IN statement ignores the setting.

EXAMPLE

CONTROL-CHARS
Control characters OFF (allowed)

CP

The jBASE CP command utilizes the jBASE copy command to output records from a file to a spooler print job.

COMMAND SYNTAX

```
CP filename {recordlist} {(Options}
```

SYNTAX ELEMENTS

filename is the name of the file from which to copy records.

recordlist is the list of record ids which to copy, the recordlist may be supplied by an active SELECT list or default to all records in file.

Options can be the following:

X display in hexadecimal format.

EXAMPLE

```
CP CUSTOMERS SMITH
```

CT

The jBASE CT command utilizes the jBASE copy command to output records from a file to the terminal.

COMMAND SYNTAX

```
CT filename {recordlist} {(Options}
```

SYNTAX ELEMENTS

filename is the name of the file from which to copy records.

recordlist is the list of record ids which to copy, the recordlist may be supplied by an active SELECT list or default to all records in file.

Options can be the following:

X display in hexadecimal format

EXAMPLE

CT CUSTOMERS SMITH

There are three commands that allow global control to the updates, through jEDI, of all database operations.

DB-PAUSE

This command will pause all database operations, through jEDI, to the jBASE database.
For example...
DB-PAUSE
DatajBASE paused at Mon Nov 25 15:59:53 2002
For READ and WRITE operations

Updates are denied also to root users

Transactions will be blocked immediately

The -a option means that administrators (e.g. root users on UNIX) can still make updates to the datajBASE.

The -r option still allows read operations on the datajBASE and therefore only pauses updates.

The -t option permits users inside a transaction to continue until it terminates the transaction by either a commit (TRANSEND) or a rollback (TRANSABORT).
You can run the command a number of times with different options. For example, you might run it with the -a option to allow root users continued access. When you are sure all your normal users are paused, you can run it again without the -a option, which will suspend the updates for all users.

DB-RESUME

There are no options to this command and must be run as root user.

For example...

```
# DB-RESUME
Database is active, resumed at Mon Nov 25 16:00:33 2002
```

When run, it resumes all operations to the database.

DB-STATUS

Run this command as any user which also shows the status of the DB-PAUSE command

For example

DB-STATUS

Database paused at Mon Nov 25 16:00:50 2002

For READ and WRITE operations

Updates are denied also to root users

Transactions will be blocked immediately

Use the -t option to display users currently inside a transaction.

The -v option is the verbose option.

Use the -w option to display all users currently suspended and waiting for a DB-RESUME.

The -V option is the very verbose option and is the equivalent of -t, -v and -w together.

EXAMPLE

DB-STATUS -V

DatajBASE paused at Mon Nov 25 16:13:36 2002

For READ and WRITE operations

Updates are denied also to root users

Existing transactions can continue until complete

Port 4 is waiting for DB-RESUME

Port 23 is waiting for DB-RESUME

Port 31 is inside a transaction

1 ports inside a transaction, 2 ports waiting for DB-RESUME

One common usage of these commands will be to split mirrors on a mirrored disk system. One of the mirrors can then continue a jBASE operation following a DB-RESUME and the other mirror, which is now quiescent, used for a fast backup using operating system specific backup tools.

Some general points on using these commands:

- If a database update is currently in operation, the command will not stop existing updates. Normally these only last a matter of milliseconds, but could be longer depending on the application. For example, the file might have a datajBASE trigger associated with it, and as that's under the control of the application, any sort of delay might occur depending on the trigger.

- It does not force any data from the kernel file cache onto disk. Hence, if you paused a database and immediately started a mirror split you would get inconsistent data and possibly corrupt files, as some of the changes to files would only be in a kernel file cache rather than disk. You should wait 1-2 minutes for your sync demon to flush updates to disk, but this is only guidance as each installation is different.

In both of the above scenarios, you must satisfy yourself that consistent flushing of the data to disk before attempting any operation that depends upon it.

- The DB-PAUSE command is extremely effective. So much of jBASE, both internal and application wise, writes to files that once it executes the DB-PAUSE command, virtually no jBASE programs will run (The DB-PAUSE, DB-RESUME and DB-STATUS commands being the notable exceptions). Please treat with extreme caution. You might like to initially use the -a option to DB-PAUSE such that root (administrator) users can continue running jBASE.

- Be wary of having jBASE commands in the login script for root user as this is always bad practice, but even more so with DB-PAUSE. Be careful, not to pause a database, and when you log back on to root to resume, the login script pauses because it is paused while running a jBASE program. It is best practice that you never put jBASE programs in the login script of root user.

- By default, the DB-PAUSE command only affects individual updates, not transactions. Therefore, while the effect of DB-PAUSE is to make the files logically correct ready for backups or splitting mirrors, the database records will not be consistent. If your applications use transaction boundaries you should use the -t option to DB-PAUSE meaning the transaction will complete and make your database logically correct. Using 'DB-STATUS -t' you can check how many (if any) users are still inside a transaction and hence able to do further data jBASE I/O and so delay and maintenance until all the users have completed their transaction.

ECHO

The jBASE ECHO command sets or resets the jBASE echo enabled flag. This enables or suppresses input character echo, when using a jBASE program.

COMMAND SYNTAX

```
ECHO {ON|OFF}
```

SYNTAX ELEMENTS

ON resume input character echo

.

OFF suppress input character echo.

EXAMPLE

```
ECHO OFF
```

ECHO-ON

The jBASE ECHO-ON command sets the jBASE echo enabled flag. This enables input character echo, when using a jBASE program.

COMMAND SYNTAX

```
ECHO-ON
```

SYNTAX ELEMENTS

None

EXAMPLE

```
ECHO-ON
```

ECHO-OFF

The jBASE ECHO-OFF command resets the jBASE echo enabled flag. This suppresses input character echo, when using a jBASE program.

COMMAND SYNTAX

ECHO-OFF

SYNTAX ELEMENTS

None

EXAMPLE

```
ECHO-OFF
```

HUSH

The jBASE HUSH command resets the jBASE echo enabled flag. This suppresses input character echo, when using a jBASE program.

COMMAND SYNTAX

```
HUSH
```

SYNTAX ELEMENTS

None

EXAMPLE

```
HUSH
```

LISTDICTS

The jBASE LISTDICTS command displays in page format the attribute definition entries in the MD file or the specified dictionary file.

COMMAND SYNTAX

```
LISTDICTS {filename}
```

SYNTAX ELEMENTS

Filename is the name of an alternative file.

EXAMPLE

```
LISTDICTS PAYROLL
```

LISTF

The jBASE LISTF command displays in page format all MD entries with attribute one set to either “D” or “Q”.

COMMAND SYNTAX

```
LISTF
```

SYNTAX ELEMENTS

None

EXAMPLE

```
LISTF
```

LISTU

Use the LISTU utility to display information on processes executing jBASE programs.

LISTU (Options

Option	Description
P	Redirect output to printer
N	No page

Calling the JBCUserCustomizeDisplay subroutine LIST-LOCKS customizes the output of the LISTU command.

Provided to display and control locks taken by jBASE applications it provides several utilities.

Command	Description
LIST-LOCKS	List all lock types.
LIST-SYSTEM-LOCKS	List system locks.
LIST-GROUP-LOCKS	List binary or group locks.
LIST-RECORD-LOCKS	List item or record locks.
LIST-EXECUTION-LOCKS	List execution locks.
SHOW-RECORD-LOCKS	Alternative display for record locks.
CLEAR-RECORD-LOCKS	Clears record locks. (Permission required)
CLEAR-BASIC-LOCKS	Clears execution locks. (Permission required)

NOTE: Execution locks have been coded, as record locks on a temporary file and so will also be displayed as record locks.

LOGON/LOGOFF

LOGON

Use the LOGON utility to logon a background process to execute a user login. The user login will execute the *.profile, which can be modified in order to execute a logon proc via the “exec jsh -“ mechanism.

LOGON Port, UserName, Password

LOGOFF

The LOGOFF utility can be used to logoff another process. Root privileges are required to logoff another process logged on with a different user id.

LOGOFF Port

list-open-files

This shows what files a user has open. By default, it shows a summary. For example

```
jsh --> list-files
Polling 3 ports ...
Opened files at 19:52:16  16 JUN 2003
Page    1
Port     Application Open Files Actual O/S Open Files
0        258                    247
1        8                      8
2        4                      4
```

NOTE: Many of the new commands get their information by polling active ports for their information. This can take 5-10 seconds.

With jBASE, 4.1 if you open the same file multiple times to different variables, then inside jBASE will have only a single operating system file. This is for efficiency and thread consideration. The difference between the numbers of files the application thinks it has open and the number of files jBASE actually has open with the operating system is displayed in the second and third columns. This will be more significant in multi-thread applications that might open the same file, once for each thread. In this case, there will only be a single actual file opened and the threads will share the same file handle, although this is done hidden from the application.
The (V) option will show a row for each file opened by the application.

MSG

The jBASE MSG command sends a message to other users using either port numbers or account/user ids.

COMMAND SYNTAX

```
MSG {!port | account/userid } text
```

SYNTAX ELEMENTS

!port is the target port number for the message.

account/userid is the target user or users for the message.

text is the message body.

EXAMPLES

```
MSG !31 Happy Birthday Jim
```

Sends a message to the "tty" device associated with port number 31

MSG Jim Can I borrow your pipe and slippers?

Sends message to the "tty" device for all users named Jim.

NOHUSH

The jBASE NOHUSH command sets the jBASE echo enabled flag. This enables input character echo, when using a jBASE program.

COMMAND SYNTAX

```
NOHUSH
```

SYNTAX ELEMENTS

None

EXAMPLE

```
NOHUSH
```

OFF

The OFF command will close down all current jBASE programs, but will stop if it encounters a non-jBASE program. To be fully effective only run jBASE programs. For example on UNIX your .profile might have:

exec MAIN.PROG

or

exec jsh

If your .profile looks like this:

MAIN.PROG

or

jsh

Keep the UNIX shell active and the effect of the OFF command will be to return you to the UNIX shell.

PHANTOM

Use PHANTOM to start a process that executes in the background. A phantom process cannot require input from the terminal.

Syntax

```
PHANTOM [BRIEF] [SQUAWK] command
```

Qualifiers

Description

Phantom process started on Process ID pid#.
The operating system assigns the process ID number, *pid#*.
A phantom process cannot use any tty devices. Input to a phantom process processes can be achieved using the Database Statements. For example, the following paragraph runs the BASIC program MYPHANTOM and supplies input to it using two
DATA statements:

```
BACKGROUND
0001: PA
0002: RUN BP MYPHANTOM
0003: DATA A
```

BRIEF Output from the phantom process is redirected to to the null device. Messages are not suppressed by the BRIEF option.
SQUAWK Displays the record ID of the record created in the &PH& file by the phantom process.
command The command to execute phantom from a process.

RUN Command

The RUN command is used to execute compiled but uncataloged jBASE BASIC programs. Whenever possible, you should catalog your programs rather than RUN them - the RUN command is only maintained for compatibility with older systems.

The format of the RUN command is as follows.

```
RUN SourceFilename ProgramName (options
```

SHOW-ITEM-LOCKS

The SHOW-ITEM-LOCKS command displays details of locked items in jBASE-hashed files.

COMMAND SYNTAX

```
SHOW-ITEM-LOCKS
```

NOTES

Calling the JBCUserCustomizeDisplay subroutine can customize the output of SHOW-ITEM-LOCKS.

Use the CLEAR-ITEM-LOCKS command to explicitly clear a lock.

SLEEP

The jBASE SLEEP command causes the process to sleep either until a specified time or for a specified number of seconds.

COMMAND SYNTAX

```
SLEEP {time|seconds}
```

SYNTAX ELEMENTS

Time specifies the time until which the process should sleep. The time value is in the 24-hour format, HH:MM:SS.

Seconds specifies the number of seconds for which the process is to sleep.

NOTES

If invoking the debugger during SLEEP and continuing execution it prompts the user:

Continue with SLEEP (Y/N)?

If "N" is the response, the SLEEP will terminate.

EXAMPLE

```
SLEEP 13:15
```
The process will sleep until 1:15 p.m.

```
SLEEP 300
```
The process will sleep for 300 seconds or 5 minutes.

SUBD

The jBASE SUBD command subtracts two decimal numbers.

COMMAND SYNTAX

```
SUBD Num1 Num2
```

SYNTAX ELEMENTS

Num decimal number

NOTES

Subtracts decimal number Num2 from decimal number Num1

EXAMPLE

SUBD 89 10

79

TERM

The jBASE TERM command enables users to specify different terminal types to handle terminal characteristics.

COMMAND SYNTAX

```
TERM {type | parameters}
```

SYNTAX ELEMENTS

Type specifies the terminal type. The type must be one of the terminal types defined in the terminfo datajBASE. If you wish to specify a known terminal type, e.g. vt220, as a different name, then achieve this by linking the new name, e.g. V, to the terminfo entry for vt220.
Specified parameters are as follows:

Parameters	Description
Tlength	terminal line length
Tdepth	terminal display lines per page
Plength	printer line length
Pdepth	printer display lines per page
reserved	Reserved for future use

NOTES

An SP-ASSIGN may result in resetting the printer line length and lines per page because of the WIDTH and DEPTH parameters associated with the specified queue's formtype.

EXAMPLE

```
TERM vt220
```
Invokes terminal characteristics for terminfo type vt220

```
TERM 132,,,,,,112,30
```
Sets the terminal line length to 132 columns; it will format any paged output to the printer into 112 columns and 30 lines per page.

TIME

The jBASE TIME command returns the system current time and date.

COMMAND SYNTAX

```
TIME
```

SYNTAX ELEMENTS

None

EXAMPLE

```
TIME
10:30:01 04 JUL 2001
```

WHERE

Use the WHERE utility to display information on processes executing jBASE programs.

```
WHERE Ports (Options
```

Where Ports can be a range of Ports of the form Port-Port

Option	Description
A	All ports displayed.
N	NOPAGE.
P	Redirect output to printer.
R	Raw output for scripts, no header, 1 line per logged on user.
S	Display processes NOT waiting at jSHELL Prompt.
U	Suppress own process from display.
V	Verbose output.

NOTES

You can customize the output of WHERE

WHO

The WHO utility displays the assigned port number and user for the current process. The user string reflects the JBCLOGNAME if configured otherwise the user name is the login user id.

WHO {PortNumber}

If specifying the PortNumber, it displays the user for that port.

For some emulations, (e.g. ros), it displays the system name and login user id.

XTD DTX

Several utilities are provided to convert, add, multiply, subtract and divide numeric decimal and hexadecimal values. Currently these are available on UNIX platforms only.

Command	Description	Example	Result
ADDD	Add decimal	ADDD 10 20	30
ADDX	Add hexadecimal	ADDX A 14	1E
SUBD	Subtract decimal	SUBD 20 10	10
SUBX	Subtract hexadecimal	SUBX 14 A	A
DIVD	Divide decimal	DIVD 20 10	2
DIVX	Divide hexadecimal	DIVX 14 A	2
MULD	Multiply decimal	MULD 20 10	200
MULX	Multiply hexadecimal	MULX 14 A	C8
DTX	Convert decimal to hex	DTX 10	A
XTD	Convert hex to decimal	XTD A	10

Comment Sheet

Please give page number and description for any errors found:

Page	Error

Please use the box below to describe any material you think is missing; describe any material which is not easily understood; enter any suggestions for improvement; provide any specific examples of how you use your system which you think would be useful to readers of this manual. Continue on a separate sheet if necessary.

Copy and paste this page to a word document and include your name address and telephone number. Email to documentation@jbase.com