TEMENOS™

jBASE Overview

What's new in release 5.

Copyright

Acknowledgements

Information regarding Unicode has been provided in part courtesy of the Unicode Consortium. The Unicode Consortium is a non-profit organization founded to develop, extend and promote use of the Unicode Standard, which specifies the representation of text in modern software products and standards. The membership of the consortium represents a broad spectrum of corporations and organizations in the computer and information processing industry. The consortium is supported financially solely through membership dues. Membership in the Unicode Consortium is open to organizations and individuals anywhere in the world who support the Unicode Standard and wish to assist in its extension and implementation.

Portions of the information included herein regarding IBM's ICU has been reprinted by permission from International Business Machines Corporation copyright 2001

Errata and Comments

If you have any comments regarding this manual or wish to report any errors in the documentation, please document them and send them to the address below:

Technical Publications Department

TEMENOS UK Limited

2 PeopleBuilding

Hemel Hempstead

Hertfordshire

HP2 4NW

England

Tel SB: +44 (0) 1442 431000

Fax +44 (0) 1442 431001

Please include your name, company, address, and telephone and fax numbers, and email address if applicable. documentation@temenos.com

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-down menu.

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.

JBASE RELEASE 5 OVERVIEW

The release of jBASE 5 undoubtedly marks a major new software development for jBASE Software. This release builds on the successful foundations laid down by release 4 that went before and delivers a new level of resilience through the introduction of the jBASE Dataguard suite. This collection of technologies allows jBASE to be truly non-stop as a technology platform.

This release of jBASE has been re-engineered to be a true 64 bit application and only 64 bit releases are supported for production use. Much of the core of jBASE remains relatively untouched to ensure stability

The major new features of the jBASE 5 releases are as follows:

- Checkpointing
- Warmstart Recovery
- Resilient Files
- Online Backup
- Resizing files
- New APIs
- 64 bit support
- SQL Support
- Native JDBC Driver

New Features

Checkpointing

JBASE 5 investment has been predominately in the area of transaction journaling which has been substantially engineered to provide an ever greater level of flexibility and robustness. A number of new structures have been introduced to allow more precision in recovery. One of the major enhancements has been the addition of a concept of Checkpointing.

Periodically, at predefined intervals, Checkpointing pauses new transaction activity to record a point in time when the database is in a known state. To record this instant in time, a checkpoint record is written to the transaction together with a timestamp.

Developments such as warmstart recovery and online backup would not have been possible to implement if it weren't for checkpoints.

Although the overhead of performing a checkpoint is minimal, the time interval between checkpoints is user configurable.

Warmstart Recovery

It has always been possible to recover a jBASE database by restoring the last backup and replaying the transaction journal but there was manual intervention required. Warmstart recovery can automate the recovery of a system that has been improperly shutdown with no user input.

This allows for automatic recovery from such things as power failures in a similar fashion to mainstream RDBMS products.

Resilient Files

In Badly sized hash files, a situation can arise where data is split across a number of different frames which all have to be read from and written to the disk when the data changes. If something went wrong during this period, there was a window of opportunity where the update could be interrupted and the structure of the file damaged. In jBASE release 5, a new type of file called a Resilient File has been introduced to eliminate this possibility.

In a resilient file, where data spans across frames on disk, it is built up in a separate area of the file and flushed to disk. Only once we know it is on disk, do we update a pointer to point to the new data structure rather than the old structure. The structure is such that all critical writes are exactly one disk

block in size which file systems guarantee to be atomic and so there is no potential for the structure of the file to be damaged in any way.

Due to the extra flushing of resilient files, there is a performance overhead when using these files.

Online Backup

Backup and restore commands have now been enhanced such that a database backup can be performed without the need to shutdown activity on the database server. The backup will be a complete copy of the database as at the point of the time when the backup process finished. This backup is guaranteed to be in a consistent state because of the checkpointing mechanism.

Resizing files

As well as being resilient, JR files also resize automatically to fit the data that is stored within them. This has been achieved by essentially divorcing the location of the data from the record key of the file. This means that the structure of the underlying data can expand without having to redistribute all the data as was the case with previous hash files.

New APIs

Application programming interfaces have been redesigned and streamlined with the introduction of a socket based protocol for communication between a client process and the server running jBASE. This eliminates the need for RMI and makes for faster communication between the two. As well as support for Java via Java OBjEX, there is also support for Dot Net applications through Dot Net OBjEX. This will allow applications written using the Dot Net framework to connect to jBASE on any supported platform.

64 Bit Support

Many operating system specific restrictions under 32 bit operating systems meant that jBASE was constantly having to work around them under the covers. With the move to a true 64 bit architecture, these restrictions no longer apply.

Native SQL Support

The query processor in jBASE has been enhanced to be able to accept SQL commands as well as jQL commands.

Native JDBC Driver

JDBC support allows external applications written in Java to retrieve data from jBASE via an industry standard interface.

Documentation

Operationally, the core of jBASE works in exactly the same way as jBASE 4.1, which means that the documentation for this release carries over untouched from the previous release.

The following new features are documented in the new manual jBASE Dataguard;

- Checkpointing
- Warmstart Recovery
- Online Backup
- Resilient Files
- Resizing Files

There is no documentation for the new release being a 64 bit compliant release as this is transparent to the operator. There is an additional manual covering OBjEX and the Dot Net API is covered in a compiled help file.

Migration Overview

Use the following steps to migrate from previous jBASE releases to jBASE 5:

1. Set up a test migration area on the server system

Physically, this involves manually creating a directory (or directories) that will hold all the data files and program source code files

2. Copy existing system file, accounts and data and source files to test area

Copy the files to the new directories. Note: BIN and LIB directories are not required, as they will be recreated in a later step. Note the location of the MD and SYSTEM files if they exist

3. Install and configure the jBASE 5 release

Both jBASE 5 and previous releases will run on the same machine, however, there are a few points to be aware of:

i. Both versions *may* share environment variables. This is only an issue if environment variables have been set up as e.g. a System or User variables on Windows. If a logon script is used on an operating system (e.g. through telnet) then the variables will be local to that instance and will not cause any problems.

ii. The main variables that a user should be aware of are 'JBCRELEASEDIR' and 'JBCGLOBALDIR'. These two variables point to the instance of jBASE (whichever version)

iii. The next variable to note is the 'PATH' variable. This variable is used to locate the executable programs (or jBASE commands). The order of the BIN directories in the 'PATH' variable will dictate the order that the system will search for the commands in

iv. Under certain circumstances, previous jBASE releases will have created a 'libjbase.dll' file in the 'Windows\system32' directory. If this is the case then the file should be moved to 'lib' to a more relevant 'lib' directory.

4. Configure a 'jbaseadm' user for system administration

This should be done at operating system level.

Note: After installation of jBASE 5, an 'Environment' script (or 'Environment.bat' on windows) is created it the 'jbase5/5.0/' directory. This simple script contains the environment variables used by

release 5, which can be copied into any logon script created for an administrator. However, for a ‘normal’ user of the account (not an administrator), the ‘iju’ facility should be used (see step 6 later)

5. Start any required processes

If jRLA / jDLS are to be used – start them using the required options.

6. Configure the server user id and set up the profile using ‘iju’ utility

The ‘iju’ utility is a command line program that creates a login script for users. Prior to running the ‘iju’ program, some users need to be set up on the operating system. Running the ‘iju’ facility will prompt for information pertaining to the user in question (e.g. user name, location of MD and SYSTEM files etc.), this information is then used to create a login script for the user in question. In the case of Windows, this will be a ‘Remote.cmd’ file. This should be placed into the relevant directory for the user. The Windows OS will automatically run this script when the user connects to via Telnet. In the case of UNIX, the script forms the basis of the users ‘.profile’ script.

7. Execute ‘jsh’

Depending on the configuration of the user, connect to the jBASE machine. In some cases this will be via telnet and run the command ‘jsh’. If the system has been set up with OS Global\System environment variables in Windows (not a telnet user) then run the command ‘jSH’ from a DOS command line

8. Logto each account and do step 11-17

Issuing the command ‘LOGTO <*accountname*>’ in the jSHELL will connect the user to the selected account that was created in step 6 above. Any variables set up in the ‘Setting’ parameters during the set up of the accounts should now be set. To test this, use ‘echo %*VARNAME*%’ (Windows) or ‘echo $*VARNAME*$’ (UNIX)

9. Update the MD/VOC, using the ‘updatemd’ utility

The ‘updatemd’ command copies information from a ‘Master MD template’ into the MD/VOC of the account. This information will be required by jBASE.

10. Optionally convert source and data for international mode using ‘jutf8’.

The command line program ‘jutf8’ is used to internationalize data. Before the program can be run, the account must have internationalisation enabled.

11. Delete all old object files, i.e. dollar records.

Search through the directory structure created in step1 and manually delete old object files. These are typically files that start with a '$' or '!' character and may end with a '.obj' or '.o' extension

12. Delete any existing 'bin' and 'lib' directories.

Search through the directory structure created in step1 and manually delete any 'bin' and 'lib' directories. These directories will hold the programs and libraries that were used and will need to be re-built in the next steps

13. Parse source files for any new key words using the 'portbas' utility.

The command line program 'portbas' is used on individual source files (one at a time) to ensure that the source code conforms to the jBASE 5 standards. The program automatically creates a back up copy of the original file and makes any changes necessary to the file.

14. Optionally create 'OBJECT' data sections for source dir/files

New functionality in jBASE allows the OBJECT files to be created in a different location to the actual source code (to prevent 'clutter'). This is accomplished by creating a 'datasection' to hold the data files.

15. Compile source files using BASIC command

The command line program 'BASIC' is the first of a two-step process to convert source code into an executable/callable form. The 'BASIC' command checks the code to ensure that it is syntactically correct. To use the command from a jSHELL, navigate to the relevant directory and call the BASIC command supplying the name of the source code:

E.g. BASIC <SourceFile> <program\sub name in the file>

Alternatively, the star character can be used to signify all programs\subroutines in the source file:

E.g. BASIC <sourceFile> *

16. Catalog source files using CATALOG command

The command line program CATALOG is run after the BASIC command has been run successfully and is the second step of the process. The command will create the executable program that can be

called, or in the case of subroutines, create a '.DLL' '.SO' file. The syntax for the CATALOG command is the same as that for the BASIC command.

17. Test Application

Test Migration Area

jBASE 5 has been designed to enable users to run previous releases of jBASE on the same machine without either release interfering with the other.

The jRLA, record locking daemon, has a different layout for jBASE 5 and is not compatible with previous jBASE releases. However, a jBASE 4.1-jRLA process is completely independent of any previous jBASE release jRLA demon processes.

By having two different jRLA, systems, users working on the same machine can work on a previous jBASE release as well as the jBASE 5 release. However, if a user takes a lock using a previous jRLA it will not be recognized by a program executing under the jBASE 5 release. Hence when testing the migration to 5, users should be sure to use a discreet set of test data such that locks taken for different releases do not cause confusion.

Migrating Code

Optionally, convert source files for International Mode using the ‘jutf8’ utility. This only applies to source intending to execute in International Mode.

Delete all old dollar object files. This step ensures that the catalog procedure does try and link elderly object files from previous jBASE releases into shared libraries.

Execute PORTBAS against all program files

Changes reserve word variables to initial caps

Insures that name of a subroutine and the name used on the SUBROUTINE statement are the same

Compile and Catalog program files

Test

Note: Utilities ‘jconvertfile’ and ‘jcompilefile’ may be of use to automate and generate reports for the PORTBAS, BASIC and CATALOG procedures.

UPDATEMD

Use the ‘updatemd’ utility on all migrated accounts as the MD/VOC entries are not compatible with previous jBASE releases. The environment variable JEDIFILENAME_MD should be set automatically according to the configuration details of the account as per the jAdminServer. If the optional jAdminServer and jExplorer have not been used then this environment variable needs to be configured to point to the MD or VOC for the account.

The ‘updatemd’ utility is used to copy the standard jBASE master dictionary entries (jQL commands and modifiers) to the master dictionary referenced by the JEDIFILENAME_MD environment variable. The ‘updatemd’ utility should only be required when upgrading from a previous jBASE release or importing new accounts into jBASE.

NOTE: the ‘updatemd’ utility will overwrite the existing MD/VOC entries with jBASE versions, hence any user entries that conflict with jBASE entries will be overwritten.