tableBASE

Programming Guide

Release 6.0.3 /
Release 6.1.0

Document Number: TBM003-R603610v2

Revision History:

Release 6.0.1—August 2004

Service Pack 2 Release 6.0.1 Version 1.1—November 2004
tableBASE Release 6.0.2 Version 1.0—May 2005
Messages revision—Release 6.0.2 Version 1.1—July 2005
Messages revision—Release 6.0.2 Version 1.2—February 2006
tableBASE Release 6.0.2 Version 1.3—March 2006
tableBASE Release 6.0.3 Version 1.0—April 2008

tableBASE Release 6.0.3 / Release 6.1.0 Version 2.0—February 2010

Technical Support Hotline: (613) 523-5588

DataKinetics Ltd.
202 - 2460 Lancaster Road
Ottawa, ON
Canada K1B 4S5

Telephone: (613) 523-5500
1-800-267-0730 (toll-free in the US and Canada)
Facsimile: (613) 523-5533
Email: tableBASE@dkl.com
http://www.dkl.com

Table of contents

Preface

This guide provides information about tableBASE commands, the tableBASE application programming interface (API), and other tools and concepts that will be of benefit to a programmer.

Who this guide is for

This guide is intended for programmers who develop batch and/or online applications that use tableBASE tables.

What you should know to use this guide

You should be familiar with a programming language, compile procedures, the linkage editor, and MVS JCL.

What is covered in this guide

The *tableBASE Programming Guide* provides tableBASE programmers with the information necessary to program and maintain the tableBASE system.

tableBASE basics

Chapter 1 provides an overview of tableBASE basics. Together with the *tableBASE Concepts and Facilities Guide*, this chapter will provide you with a basic understanding of tableBASE.

How to call tableBASE

Chapters 2 through 6 describe the calls to tableBASE. This section also includes examples and highlights operational differences for the various environmental interfaces.

Chapter 2 “tableBASE basics” provides tableBASE operations overview and is intended for frequent reference. The *tableBASE Quick Reference Guide* includes much of the content from this chapter.

Chapter 3 “tableBASE commands” describes how each tableBASE command works.

Chapter 4 “tableBASE parameter description” provides an alphabetical list of all the parameters used when executing tableBASE functions.

Chapter 5 “tableBASE coding examples” provides coding examples that document a number of recommended programming conventions for tableBASE applications, including naming conventions and suggested coding techniques. tableBASE coding examples are provided in C, COBOL, and Assembler.

Chapter 6 “Environmental interfaces” contains additional information regarding JCL and the variations with each environment and interface.

Operational considerations

Chapter 7 “Customizing tableBASE options” discusses operational issues that affect your use of tableBASE. In particular, it focuses on run-time parameters which allow you to adjust the functioning of tableBASE.

Accessing tableBASE

Chapter 8 “tableBASE driver command processor for CICS” discusses the TBDRIVC (DK1TDRVC) command processor for the tableBASE/CICS interface.

Chapter 9 “TBDRIVER command processor for Batch/TSO” discusses the TBDRIVER (DK1TDRV) command processor for the batch interface.

These command processors can be used in a variety of ways—for example, application development and testing, learning how to make tableBASE calls, simulating and verifying tableBASE command sequences, or monitoring the usage and resource requirements of tableBASE.

Diagnosing errors

Chapter 10 “Error diagnosis” is intended for the analyst with skills in examining storage dumps. Details of diagnostic information and analysis approaches are provided to help determine the cause of errors.

tableBASE subroutines

Chapter 11 “tableBASE subroutines” is intended for programmers who are interested in using lower-level data access routines. This is usually done when one of two conditions occur:

- The data relationships are very stable and can be easily managed in programmer controlled storage.

- The access speed is critical and a shorter instruction path length is required.

Use of the TBACC and TBINDX routines should be restricted to personnel who manage program storage areas.

In addition, for those programmers processing the contents of a View, information on the subroutine DKTBNAME, which provides View Name conversion, is also provided.

User exits from tablesONLINE

Chapter 12 "tablesONLINE/ISPF exit programming" and Chapter 13 "tablesONLINE/ CICS exit programming"contain specifications and tutorial information for coding tablesONLINE User Exits (tablesONLINE is an optional component of tableBASE). Examples are provided. This is intended for people who wish to integrate specialized processing into tablesONLINE for the entire system or for particular tables.

Supplementary material

The appendixes provide supplementary material to aid in your use of tableBASE.

- Appendix A on page 377 contains a number of hints and suggestions to obtain the maximum benefits from tableBASE. It includes suggestions for optimizing performance.
- Appendix B on page 383 describes the enhancements made to tableBASE beginning with Release 5.0 and explains how Release 6.0 maintains compatibility with previous releases.
- Appendix C on page 389 describes tableBASE run-time overrides.
- Appendix D on page 405 describes TBOPT dataset coding.
- Appendix E on page 407 provides a list of error codes and messages.

What's new in Version 6

The *tableBASE Release Notes* provide a thorough understanding of all of the enhancements included in this release of the product. Programmers should be aware of the following modifications:

- tableBASE Version 6 provides a fully re-entrant engine that extends tableBASE performance benefits to multi-threaded applications.
- Virtual Table Share—Virtual tableSPACE Region (VTS-TSR) provides read and write access to shared tables from any region.

  Note: VTS is a shared TSR and will be referred to in this document as VTS-TSR to distinguish it from the local (private) TSR. The term TSR when not qualified refers to either a local or a VTS-TSR.

- tableBASE Process Manager—A three-tiered management application that provides flexible control of VTS-TSRs, and provides automation for VTS-TSR change control.
- tableBASE uses Data Spaces for local TSR and VTS-TSR storage allowing for an increased TSR size up to 2G.
- Version 6 provides a single unified API that supports all stubs from prior releases, for all environments, with a single link-edit "include" library.

  Note: VTS users who run applications that employ the TBCALLV and TBASEV interfaces should refer to "VTS User access" on page 239for additional information.
- Locking of objects is significantly improved in Version 6. The granularity of locking is at the table-level rather than the entire TSR.
- Many code paths have been shortened to provide faster, more efficient access to tables. This is particularly true of access to VTS-TSR tables where the code path is now the same as that for the local TSR.
- The tableBASE Root and Nucleus modules are above the line, as is the tableBASE CICS Resource Manager.
- There is special handling of DB2 Stored Procedures that allows the use of tableBASE to be extended to the DB2 area.
- TBOPT parameters have been added to provide runtime overrides for most of the installation options.
- Version 6 I/O routines tolerate open failures and I/O errors that may have resulted in region failures using previous versions of tableBASE.
- Improved diagnostics are provided with the use of error subcodes in the extended command area (see Appendix E "tableBASE Messages" on page 407).
- Individual libraries may be set to read-only.
- The Version 6 stub is re-entrant, allowing you to build fully re-entrant application load modules.
- Any library can be expanded.
- All API stubs are shown in CICS EDF tracing. In previous releases, TBCALLC, TBASEV, TBCALLV were not captured by the EDF trace.
- If a CICS task is put into a wait for a table exclusively held (opened for write) by another region, a CEMT INQUIRE task will show TBLBASE in the Suspend Value field.
- STGPROT may be used with TBASEV and TBCALLV.
- Since all the API stubs are re-entrant, they may be used to let applications take advantage of the CICS RENTPGM feature.
- Support for CICS storage protection keys is enhanced. Almost all the code run while in tableBASE modules is in the Exec key set by the installation's CICS EXECKEY definitions, thus affording the protection implied by this CICS feature.

- Various enhancements to Strobe are included, such as the production of finished reports directly in CICS.
- The CICS TBOPT file may now be QSAM (including DD *) or, as it was in previous releases, VSAM.
- More flexibility in specifying TBOPT input, more options, and the ability to list all options have been added.
- TBOPTV is merged with TBOPT, allowing for a single source of run-time parameter input.
- The allocation of VTS-TSRs is done in a way that does not impinge upon your site's MVS MAXCAD parameter.
- The default data-sensitive processing date rolls over at midnight.

Naming protocol

Version 6 features the new tableBASE naming protocol. All tableBASE executables begin with DK1 for easy identification, a prefix that has been reserved for our exclusive use with IBM.

Aliases are retained for the API (TBLBASE) and batch utilities so that no changes are required to your existing applications. For CICS, new definitions are included to update the CDS.

New modules should be used to entirely replace earlier Version 6 modules.

Conventions Used in this Guide

This guide uses conventions to differentiate code and typed commands, to display the names of parameters, and to describe specific version and release levels of the tableBASE product.

Convention	Description
`code examples and commands`	Code examples and commands appear in this type of font: `this is an example of the font.`
MAXNMTAB	Names of parameters appear in upper case simply for ease of reading; actual case used is upper or lower or a mixture.
Version	Following IBM standards, the term *version* refers to a generation of a software product that has significant new code or new functionality. *Version* is a more general term than *release*. For example, *Version 6* includes *Release 6.1* and *Release 6.2,* and is equivalent to *Release 6.x*.
Release	Following IBM standards, the term *release* refers to a program or set of programs which represent a specific revision to the base version of a software product. For example, *Release 6.0* is a term that is used to identify the first release of *Version 6*. Subsequent releases made available under the Version 6 umbrella, such as *Release 6.1*, will provide additional revisions to the base product.
Modification Level	Following IBM standards, the term *modification level* refers to the application of specific program enhancements and error corrections to the release of a software product. For example, *Release 6.0.3* is at *modification level* 3, and *Release 6.1.0* is at *modification level* 0.
MVS	MVS is a generic term which is used when referring to z/OS and other related IBM operating systems.

Conventions for entering parameters

[ ] Indicates optional selection of parameters. Do not enter the brackets.

Optional parameters are positional, i.e., when two items exist in the brackets both parameters must be entered. For example, the notation "[PARMA [PARMB]]" means that if PARMB is given then PARMA must also be given as a place-holder in the argument list. If you want PARMA to be ignored, then set it to nulls or spaces, as appropriate for the command.

For a complete description of all tableBASE parameters, see Chapter 4 “tableBASE parameter description” on page 141.

The "xxxx" in xxxx-COMMAND-AREA is a programmer-defined variable prefix in the application program code. xxxx-COMMAND-AREA may also be referred to as COMMAND-AREA in this manual.

Additional tableBASE references

This guide is one of a series that describes tableBASE and tablesONLINE:

- *tableBASE Release Notes*
- *tableBASE Installation Guide*
- *tableBASE Concepts and Facilities Guide*
- *tableBASE Batch Utilities Guide*
- *tableBASE Administration Guide*
- *tableBASE Programming Guide*
- *tableBASE Quick Reference Guide*
- *tablesONLINE/CICS User's Guide*
- *tablesONLINE/ISPF User's Guide*

Glossary

Data Table	A Data Table is the actual raw data. Each Data Table has a table definition (DT-BLOCK) that is used to generate the Index for the Data Table.
Index	An Index is defined for each Data Table. A Data Table Index is generated dynamically when a table is opened or defined based on the information in the table definition (DT-BLOCK).
Alternate Index	An Alternate Index is an Index that may be defined for a Data Table. The Alternate Index has an Alternate Index definition (ALT-DEFINITION) that defines the key, organization, and search order. Alternate Indexes are optional, and there is no limit to the number of Alternate Indexes a Data Table may have.
Delivered defaults	The defaults that are delivered with the product. Also known as *factory defaults*.
Installation defaults	The defaults set at installation time by an administrator, which may or may not be the same as the delivered defaults. Defined using the TBOPTGEN file. (These defaults may be overridden by an individual application using the TBOPT file.)

TSR	Table Space Region. A data space of up to 2G is used by tableBASE to house tables. The data space is owned by an application in the associated address space. The application uses tableBASE to access data within the tables.
Local TSR	As above.
Shared TSR	A VTS-TSR; see below.
Temporary Table	A temporary table exists only within a TSR, and is created by the DT command (or IA). It is never stored in a library. A temporary table can be distinguished from a library table using the GD command output—if found, a temporary table will show no dataset name.
Linked Table	A linked table (also known as a remote table) is created when a user issues a command to open a table that is already open in a VTS-TSR specified in the LIB-LIST. The table entry in the local TSR is linked to the existing open table in the VTS-TSR. No updates are allowed to a linked table.
Table Expansion	Dynamic allocation of space for tables in the TSR when the initial space allocated becomes insufficient.
Multitasking Batch	An MVS region that implements multitasking by attaching multiple TCBs. This can include a batch job that attaches several subtasks or a transaction processing region like DB2 stored procedures that implements multitasking through multiple TCBs.
View	A tablesONLINE View provides the field, edit and display attributes for a Data Table with its Index. In releases previous to Version 6 (and Version 5) the View was referred to as a Field Definition Table (FDT).
Alternate Index View	A tablesONLINE Alternate Index View is identical to a View but applies to a Data Table when access is through an Alternate Index.
VTS	Virtual Table Share. A VTS-TSR is a shared Data Space that provides a public location to share tables among application regions.
tableBASE VTS	The optional component that provides the shared VTS-TSR capability.

VTS-TSR	A Virtual Table Share (VTS) Table Space Region (TSR) is a shared TSR, and resides in a shared data space. Applications can access tables within a VTS-TSR, and use the information as if it were within their local TSRs. VTS-TSRs are managed by the VTS Agent program. If tableBASE Process Manager is installed, VTS-TSRs are managed by VTS Managers, and the VTS Agent is not required but still available for transition purposes.
tableBASE Process Manager	The optional component that extends the functionality of the tableBASE VTS shared TSR capabilities.
TPM	tableBASE Process Manager—the tableBASE Process Manager top tier component.
VTS Manager	The middle tier of tableBASE Process Manager which manages VTS-TSRs.
TPVM	This is synonymous to a VTS Manager. It is the middle tier of tableBASE Process Manager which manages VTS-TSRs.
Catalog	A catalog contains definitions of managed items in the next tier down in the hierarchy; i.e., TPVM definition information is contained in the TPM catalog, and VTS-TSR definition information is contained in the TPVM catalog. The catalog is contained within the LDS associated with the TPM / TPVM.
Cataloged VTS	A VTS-TSR that is managed by a user-defined VTS Manager under tableBASE Process Manager, as opposed to the *compat* VTS Manager.
LDS	Linear DataSet used for tableBASE Process Manager.
Alias name	An alternate name for a VTS-TSR that can be created, assigned, and used in lieu of the name assigned at VTS-TSR definition.
VTS switch	The action of switching an alias name from one VTS-TSR to another. This is a feature of tableBASE Process Manager.
***compat* VTS Manager**	The *compat* VTS Manager is the default VTS Manager that runs under tableBASE Process Manager.

1 Introduction

tableBASE is a memory-resident table-management system that handles row data in much the same way as a database management system (DBMS) but with substantially faster performance.

With tableBASE, all table data is loaded into memory in a tableSPACE region (TSR) on the first table access, thereby making the entire table available. This permits extremely fast access to all the data and ensures that all requisite data manipulations can be performed directly in memory. tableBASE uses IBM's Data Space architecture, allowing tables to occupy up to 2G of data in virtual memory. Large table sorting and manipulation occurs in memory, without resorting to external files. Since data manipulation is done entirely in memory using very short path lengths, tableBASE avoids the I/O and CPU overheads associated with other technologies. tableBASE works with all versions of z/OS. It can be used with applications written in C, C++, COBOL, Assembler, PL/1, and any other language that can make a standard IBM call.

tableBASE improves the performance of applications that use data with a high read-to-write ratio such as validation tables, reference tables and business rules. tableBASE also facilitates the creation of on-the-fly tables that can be used to amalgamate report information or discern business intelligence information.

tableBASE provides proprietary facilities for:

- dynamic allocation of memory
- dynamic creation of indexes
- indirect accessing of tables
- date-sensitive processing
- memory management.

For more information on tableBASE, see the *tableBASE Concepts and Facilities Guide*.

This section is designed to provide the background you, as a programmer, should have to make effective use of the capabilities of tableBASE.

The base product

The basic tableBASE component, provided to all clients, is the batch interface and local TSR, with which users can build applications to work in batch or TSO. Optional tableBASE components include:

- IMS TM interface
- CICS interface
- tablesONLINE/CICS
- tablesONLINE/ISPF
- VTS (Virtual Table Share)
- tableBASE Process Manager

tableBASE terminology

Programmers who are new to tableBASE should be aware of the following concepts and unique tableBASE terminology.

tableSPACE Region (TSR)

The TSR is a Data Space used by tableBASE to hold tables in memory. The term "local TSR" is used to indicate that it is held in a private Data Space. The local TSR is a standard component of tableBASE.

Note: The term "TSR" is used to indicate a Data Space used to hold tableBASE tables. A TSR may either be local TSR (private) or a VTS-TSR (public).

Virtual Table Share (VTS)

VTS is an optional component of tableBASE. A VTS-TSR is a shared Data Space that provides a public location to share tables among application regions.

Program Call server (PC Server)

All interfaces require that the PC Server be started and running before tableBASE can be initialized in a region. It is also used to execute multitasking in batch and for use with the VTS-TSR. The PC Server must execute from an APF-authorized library and must be running prior to the start of a batch multitasking job (including DB2 Stored Procedures) or start-up of a VTS-TSR. The PC Server provides base-level VTS-TSR functionality. tableBASE Process Manager provides full-featured VTS-TSR management functionality.

Note: If you are running the optional tableBASE Process Manager product, PC Server is not needed. All of the services otherwise provided by PC Server will be provided by the tableBASE Process Manager.

tableBASE Process Manager

The tableBASE Process Manager is an optional component. It is required if you want to use its powerful management features for the control and automation of VTS-TSR change control. tableBASE Process Manager must execute from an APF-authorized library. If you use tableBASE Process Manager, it must be running prior to the start of a batch multitasking job (including DB2 Stored Procedures) or start-up of a VTS-TSR. tableBASE Process Manager supersedes the PC server functionality.

Tables

tableBASE tables are a collection of fixed-length data rows. Each row contains a key and unstructured data.

tableBASE tables provide a great performance advantage over disk-based tables for many types of data.

Applications such as tablesONLINE that access tableBASE tables can apply the column definitions to the table data. In tablesONLINE the column definitions themselves (or metadata) are contained in tables called Views.

Besides data rows, a tableBASE table also includes a table definition that defines such attributes as row length, key location, key size, organization, and search method.

Although tableBASE tables are usually loaded from and ultimately stored on a hard disk, tableBASE tables are entirely resident in memory during processing. All memory-resident tables have indexes that are created dynamically when loaded from the library (see "Indexes and Alternate Indexes" on page 33). Table data is compact; however, the table index(es) may not be (see "Table organization" on page 32).

tableBASE tables can be created in three ways:

- programmatically through a call to tableBASE
- online using tablesONLINE
- using utilities supplied with tableBASE.

In this guide we describe creating tables programmatically. You may create tables that exist only in memory for the duration of a particular job (temporary tables) or you may create tables that reside permanently in a tableBASE library on disk.

Table names

Each table that is stored in a tableBASE library must have a unique name. A valid table name is a string of 8 bytes that are not all spaces, all low values, all high values, or :TMPNAME.

There are certain classes of table names which are normally reserved for use by tablesONLINE. These names begin with a special character or lower-case alphabetic. For this reason, user-defined Data Table names should begin with an upper-case alphabetic; the remaining characters may be alphabetic, numeric, or special symbols. Although tableBASE will correctly process table names containing embedded spaces, e.g., "ABC 12", these names are not recommended. In some utilities, e.g. TBCOMP, such table names must have quotation marks around them.

Linked tables

A linked table (also known as a "remote table") is created when a user issues a command to open a table that is already open in a VTS-TSR specified in the LIB-LIST. The table entry in the local TSR is linked to the existing open table in the VTS-TSR. The table data is not duplicated in the local TSR. Use of VTSFIRST, VTSLAST, or the ML command to access a table in VTS-TSR (linked table) does not allow for update commands against the table. Updates of the table can only be achieved by the use of the VTSNAME in the TB-PARM to access the table.

Note: The use of linked tables has limitations. Most significantly, if you open a linked table within your local TSR, there will be no indication to you in cases where the actual table is changed, deleted or replaced.

Table organization

Since tableBASE operates in-memory, it offers table organization options optimized for in-memory use. Tables can be organized in any of four ways:

- Sequential
- Hash
- Random
- User-controlled sequence

Sequential

Sequential tables are ordered by ascending or descending sequential keys.

Sequential tables can be searched using a Queued sequential search, Binary search or Address tree binary search.

Hash

Hash tables are sequenced using a hashing algorithm that determines the location of a row of data in the table. Rows are stored according to a randomized function of the key. This randomizing routine ensures that rows are spread uniformly throughout the table and not heavily clustered in any particular area. Note that the table itself is compact; it is the index that has the uniform distribution.

Hash tables must be searched using a Hash search.

Random

There is no particular sequence to the table. New rows are inserted at the end of the table. In releases prior to Version 6, deletions caused the last row to be moved into the empty space. In Version 6, there is no such movement into the empty space, and we simply assume that the entries are in random order.

A Serial search is the only practical search method for this organization.

User-controlled sequence

Rows are stored in a random-like table where the sequence is controlled by the application program. The location of insertion of a new row is controlled by the COUNT field. If inserting a row using a key, the insertion is at the end of the table.

Like a random table, the sequence of rows for a User-controlled table is not predictable from data field values, and a Serial search is the only practical search method for this organization.

Indexes and Alternate Indexes

Table Indexes are generated dynamically when tables are opened, and are based on the table definition information.

The tableBASE software allows you to create multiple Indexes on any Data Table. These are referred to as Alternate Indexes. Alternate Indexes allow you to access the same data in multiple ways, using different keys.

Search methods

The tableBASE software offers you a variety of search methods that are optimized for performance, table organization, and in-memory use:

- Serial search
- Queued sequential search
- Binary search
- Address tree binary search
- Hash search

The search is applied to the Index and the Index then points to the appropriate row of the Data Table.

Serial search

A serial search compares the search key with the key of each row in the Index. The search begins at top of the Index and progresses through the Index until the row is found that contains the search key or until all rows in the Index have been examined.

A serial search uses little overhead and is the fastest search method for small Indexes. It is also more efficient for user-controlled Indexes with a skewed frequency of hits. The user can place the most frequently accessed rows at the top of the Index.

The table organization must be Random or User-controlled sequence.

Queued sequential search

This search method is executed by progressing serially through the Index and comparing the search key to each Index key until a row is found. Subsequent searches begin where the last row was compared, if the search key is farther along in the Index sequence. If the key of the next row examined is too high (or too low for descending sequential Indexes), the search resets to start from the beginning of the Index.

Queued sequential searches are faster on partially-sequenced or mostly-sequenced data—for example, transaction matching in a master file type of update process.

The table organization must be ascending or descending Sequential.

Binary search

The search key is compared with the key in the middle of the Index and determined to belong to either the top or bottom half of the Index. It is then compared to the middle of the appropriate half of the Index. This process of splitting the remaining Index rows in half continues after each comparison until the desired row is found.

The table organization must be ascending or descending Sequential.

Address tree binary search

The address tree binary search process compares the search key to the endpoints of an Index to determine if the search key is within the Index range. If the search key is not within the range, then the system returns a "not found" message. If the search key is within the range, then a binary search process is used to find the key position.

An address tree binary search is a fast technique for performing inserts into ordered data.

The table organization must be ascending or descending Sequential.

Hash search

This searching technique randomizes the key to calculate the location of a row in the table. A hash search is the best search method for large tables that are usually accessed randomly. Performance is enhanced because there is no search looping, however this creates greater software overhead because of the calculation required for the search.

Under normal circumstances hash searches and hash-based insertions use less CPU time than binary, serial, and most sequential searches, and require fewer page accesses than any other search.

Space utilization is minimized by a Hash Pointer Table—the actual data is kept in a densely packed random table, while the more sparsely packed Index contains only the address of the data rows.

The table organization must be Hash.

Search summary

Table 1-1 summarizes the valid combinations of table organizations and search methods allowed by tableBASE.

Table 1-1: Search summary

Table organization	Search methods				
	Queued sequential	Address tree binary	Binary	Serial	Hash
Sequential	Y	Y	Y		
Descending sequential	Y	Y	Y		
User-controlled sequence				Y	
Random				Y	
Hash					Y

Libraries

tableBASE tables are processed in memory, but normally they are loaded into memory from disk-resident libraries, to which they may be stored after changes have been applied to the memory-resident version. A tableBASE library is a specialized dataset initialized in a special format by tableBASE. Each tableBASE library contains a directory of all the tables in the library, and the contents and definition for each table.

A company can utilize any number of libraries. To distinguish the libraries, each is given a name by which it may be referenced in an application program. The name used is the DDNAME which allocates the library dataset in the JCL for the application (or, in the

case of CICS, the name of the CICS File definition that defines the library file to CICS). The delivered default library is MAINLIB. Additional libraries are defined through a call to tableBASE or by using the TBEXEC (DK1TEXEC) utility.

Libraries are allocated as VSAM, BDAM, or physical sequential (which will be used as if BDAM). Libraries are managed by the batch utility TBEXEC, or with tablesONLINE.

Library search order and uniqueness of table names

There may be many tableBASE libraries in existence. It is possible that more than one table may exist with the same name in different libraries. These tables may have completely different significance and contents. When a table is loaded into memory the named table is searched in all libraries where it might live (on the current LIB-LIST) by the application providing tableBASE with an ordered list of library names (LIB-LIST). tableBASE starts searching for the named table in the first library named in the LIB-LIST, then the second, and so on until the table is found or all libraries have been examined. If two tables with the same name, say, XXX, exist in more than one library, and the application requests a table named XXX is to be loaded into memory, the XXX table from the earliest library in the LIB-LIST will be loaded.

Generations

A tableBASE library will hold up to nine generations of each table, as specified in the table definition. For more information see "GENERATION (fullword binary)" on page 157.

Protecting tableBASE tables

tableBASE tables can be protected in the library (using read and write passwords) and when loaded in memory (using the LOCK-LATCH password feature). These passwords offer only limited protection against unauthorized use of tableBASE tables. More formal security can be implemented via user exits and third party measures such as Resource Access Control Facility (RACF), eTrust CA-ACF2 Security for z/OS, and eTrust CA-Top Secret Security for z/OS. Such data security systems may be used to protect tableBASE libraries. If invalid access to a library is attempted via application use of tableBASE, the access will be refused.

In the event that a table password or LOCK-LATCH password is lost or forgotten, please notify your tableBASE administrator.

Read and write passwords

Passwords are used when opening tables from the library. Once in memory (i.e., in a TSR), the tables can be accessed by any application with access to that TSR or VTS-TSR. A read password protects a table from being opened for either read or write. A write password protects a table from being opened for write. For more information on passwords see "5. RPSWD (8 bytes)" on page 151, and "6. WPSWD (8 bytes)" on page 151.

LOCK-LATCH

LOCK-LATCH passwords are used by applications to protect a tableBASE table from being changed in memory by unauthorized users in a multi-user environment. To control the user(s) or transactions that perform the subsequent updates in a multi-user environment after a table is opened for write, place a password in the LOCK-LATCH field when Opening a table for Write. Only the update commands that have the same password in the LOCK-LATCH field are authorized to perform the update. LOCK-LATCH is needed for Read-Write VTS-TSRs, but not Read-Only VTS-TSRs (available only with the tableBASE Process Manager optional component).

For further information, see "9. LOCK-LATCH (8 bytes)" on page 146.

2 tableBASE basics

This chapter provides a quick reference for tableBASE command usage. This overview serves as an introduction to the next two chapters, which provide more detailed information.

Getting started with tableBASE

The tableBASE system is accessed through the application programming interface (API) called TBLBASE. In tableBASE Version 5 and newer, TBLBASE can be used in CICS, batch, IMS, and VTS-TSR operating environments. tableBASE programs that use this stub can be ported from one environment to another without requiring any stub relinking. Of course, programs must continue to conform to the requirements of each execution environment.

The Version 6 stub, TBLBASE, is distinct from the stub of the same name used in Version 5. The TBLBASE stubs of previous releases were held in physically separate libraries—one for each operating environment. With Version 6, there is only one TBLBASE for all operating environments.

Calling protocol

Standard IBM protocols are used to invoke tableBASE. Two parameters are common to all tableBASE calls:

- TB-PARM passes environmental and release-specific information to tableBASE. For more information see “TB-PARM (64 bytes)” on page 168.
- COMMAND-AREA provides key information to tableBASE such as the name of the table to be acted upon, the action to perform, and parameters to use. The xxxx-COMMAND-AREA is dynamic; normally, parts of it are updated with each call to tableBASE. For more information see “COMMAND-AREA (72 bytes)” on page 143.

Note: The "xxxx" in xxxx-COMMAND-AREA is a programmer-defined variable prefix in the application program code. xxxx-COMMAND-AREA may also be referred to as COMMAND-AREA in this manual.

Other parameters may be required by specific commands, for example, row data to be given to TBLBASE or optional data items used by the commands.

The last parameter must be marked with a high-order bit, and must be set. This is automatic in COBOL, but must be done explicitly in Assembler. It may also need to be explicitly set in C and other languages.

In addition to defining data structures to house the required and optional parameters (including those mentioned above), if you are using retrieval or update commands, space needs to be allocated for at least one row of the table data.

The general form of a C call to invoke the common tableBASE interface is:

```
#pragma linkage( TBLBASE, OS )
TBLBASE( TB-PARM, xxxx-COMMAND-AREA, parameter-1, parameter-2);
```

The general form of a C++ call to invoke the common tableBASE interface is:

```
extern "OS" void TBLBASE( void * TB-PARM, void * xxxx-COMMAND-AREA, void *
                          parameter-1, void * parameter-2 );
TBLBASE( TB-PARM, xxxx-COMMAND-AREA, parameter-1, parameter-2);
```

The general form of a COBOL call to invoke the common tableBASE interface is:

```
      CALL 'TBLBASE' USING TB-PARM
                           xxxx-COMMAND-AREA
                           parameter-1
                           parameter-2.
```

The general form of an ASSEMBLER call to invoke the common tableBASE interface is:

```
Call  tblBase,(TBparm,Command_Area,Row_Area),VL
```

References to entities with variable length names such as libraries or VTS-TSRs must contain eight characters, whether the actual name contains eight characters or not.

Requirements for parameter-1 and parameter-2 vary, depending on the type of command to be processed. For a full explanation of parameters, see Chapter 4 “tableBASE parameter description” on page 141.

Using tableBASE

A typical use of tableBASE in your program is as follows: open the table, retrieve and/or update rows in the table, store the table to disk if you wish to save any changes made, close the table.

When a table is opened, storage is dynamically allocated in a TSR managed by tableBASE, and a copy of the table is loaded from the table library. A table may be opened for write in only one region or job at a time, but a table can be opened for read and shared by multiple regions or tasks.

The data within a tableBASE table is made available to the application one row at a time. The application provides the storage for tableBASE parameters; e.g., command areas, row areas, etc.

If the table size increases as a result of inserting new rows, tableBASE automatically expands the table.

All tableBASE processing is done in memory, using TSRs managed by tableBASE.

All changes to the table's contents are made to the in-memory version. If changes to the table are to be made permanent, the table must be written back to the library. This is done with a Store command (see "Store Table (ST)" on page 134).

Defining a table

Before a table can be used, it must be defined. You can do so online using tablesONLINE (or the TBEXEC or TBDRIVER utilities), or you may define a table in your program via tableBASE. In this latter case, you actually specify the definition in a data structure known as a DEFINITION-BLOCK and then tell tableBASE to create the table (albeit an empty table) by use of the DT (Define Table) command:

```
MOVE 'TESTTBL' TO NEW-TABLE-NAME.
MOVE 'DT'      TO NEW-TABLE-COMMAND.
CALL 'TBLBASE' USING    TB-PARM
                        NEW-TABLE-COMMAND-AREA
                        DEFINITION-BLOCK.
```

Figure 2-1: Defining a table

In the DEFINITION-BLOCK you supply such information as table organization, search method, passwords, row size, and key location. The table name is taken from the COMMAND-AREA. For the best performance, it is advisable to use a separate COMMAND-AREA for each open table.

Opening a table

This section details the codes for opening tableBASE tables.

Open for Read (OR)

If a table is opened for read, it may not be subsequently stored. It may be updated in memory when the program is running in batch. If the table to be opened has a read or write password then a password will be required to complete this command. For more information see "Open for Read (OR)" on page 121.

Open for Write (OW)

A table opened for write may be stored. If the table to be opened has a write password then a password will be required to complete this command. No other program operating in any region of any MVS image sharing the tableBASE library will be allowed to open this table for write, until the program that issued the open for write closes or releases the table. For more information see "Open for Write (OW)" on page 123, "Close Table (CL)" on page 69, and "Release Table (RL)" on page 129.

OPEN STATUS

A table which is opened will have assigned to it an OPEN STATUS in the DEFINITION-BLOCK (page 149). Before executing a command, tableBASE checks the open status of the table. Some tableBASE commands require the table to be open, while others require it to be closed. Some commands will cause the table to be opened for read automatically if it is not already open, as in the following section, "Implicit table opens".

Implicit table opens

Unlike most databases, tableBASE allows for implicit table opens. A table may be implicitly opened if a retrieval is issued for a table that is not already open and for which no read password is required. With an implicit open, tableBASE will attempt to automatically open the most recent generation of the table for read.

The attempt will be successful, only if:

1. No read password is required

2. The Implicit Open facility has not been disabled at installation time, by TBOPT Switch settings, or through the use of a Change Status command.

Implicit opening of a table is performed, subject to these restrictions, by the following commands: SK, FK, GF, GN, GP, GL, FC, FG, FN, CN and DU.

While an explicit open of a table zeroes the count field in the Command Area for all commands, an implicit open does not zero it for the FC, IC, RC, DC, and DU commands.

Note: A closed table will be implicitly opened for read by any of the following commands: IC, DC, RC, IK, DK, RK, MT, and CD. Tables that are opened for read, such as those opened implicitly, cannot be stored. To store the table, it must be opened for write.

Although the implicit table open facility is often useful, in general, it is good practice to explicitly open tables. By doing this, the code that processes error conditions for an open failure can be isolated and need not be repeated for subsequent retrieval or updating operations.

The implicit table open feature may be permanently disabled when tableBASE is installed, by use of the TBOPTGEN facility, or for a region by use of the TBOPT parameter, or temporarily disabled with the Change Status (CS) command.

In a multi-tasking environment, however, it is recommended that you do not disable this feature. In a multi-tasking or multi-user online environment, a table could be opened for read by one user and subsequently opened for read and closed by another user (who is not aware of the first). In such a situation, leaving the implicit open feature on will ensure that the table be automatically re-opened when the original user executes the next retrieval command.

Populating tableBASE tables

tableBASE tables can be populated using sequential files extracted from DB2, Oracle, etc. by using a batch import utility such as TBEXEC. For more information see the *tableBASE Batch Utilities Guide*.

A table can also be populated from within an application. Figure 2-2 is an example of opening a table for write and Figure 2-3 is an example of inserting rows into the table.

```
MOVE 'OW'          TO NEW-TABLE-COMMAND.
MOVE WRITE-PWD     TO PASSWORD.
CALL 'TBLBASE'     USING    TB-PARM
                            NEW-TABLE-COMMAND-AREA
                            PASSWORD.
```

Figure 2-2: Opening a table

The parameter PASSWORD is needed in this example as the existing table has a write password.

Assume there is a table with ten rows, each of which is defined in an array in memory called INPUT-DATA, and that the table is open for write. Figure 2-3 is an example using the Insert by Key (IK) command to load these ten rows into the table. Note that for brevity the error code and found code checking which should follow each tableBASE command have been omitted from this example.

```
           MOVE 1        TO SUBSCRIPT.
           MOVE 'IK'     TO NEW-TABLE-COMMAND.
           PERFORM UNTIL SUBSCRIPT > 10
               MOVE INPUT-DATA (SUBSCRIPT) TO NEW-TABLE-ROW-AREA
               CALL 'TBLBASE'    USING    TB-PARM
                                          NEW-TABLE-COMMAND-AREA
                                          NEW-TABLE-ROW-AREA
               ADD 1 TO SUBSCRIPT
           END-PERFORM.

      *    STORE THE TABLE

           MOVE 'ST'         TO NEW-TABLE-COMMAND.
           CALL 'TBLBASE'    USING  TB-PARM
                                    NEW-TABLE-COMMAND-AREA.

      *    NOW CLOSE THE TABLE

           MOVE 'CL'         TO NEW-TABLE-COMMAND.
           CALL 'TBLBASE'    USING  TB-PARM
```

Figure 2-3: Populating a table

Retrieval and update processing

Retrieval commands will return a copy of a single row from the table to an application's ROW-AREA. The length of the ROW-AREA may be less than the length of a row in the table. The ROW-OVERRIDE-LENGTH field of the COMMAND-AREA determines the number of bytes returned in the ROW-AREA.

Update commands will delete, replace, or insert a single row in the table. Data used for replace and insert commands resides in an application ROW-AREA. The length of the ROW-AREA may be less than the length of a row in the table. The number of bytes in ROW-AREA can be overridden by the ROW-OVERRIDE-LENGTH field of the COMMAND-AREA. The length of a row in the table is determined by the RSZ field of the table definition.

Note: For IK and RK commands, if the ROW-OVERRIDE-LENGTH is less than RSZ in the table, then the first ROW-OVERRIDE-LENGTH bytes of the row are replaced, and the remaining bytes in the table row are filled with low values.

Accessing a table

A table is accessed for four primary reasons:

- inserting a row
- reading a row
- updating a row
- deleting a row.

Inserting a row was discussed above. Reading a row is equally straightforward. A row can be read by key or by row number. Or, you may cycle through a table from row to row. You may even read rows using a partial key, so that, for example, you may read all rows whose key starts with PAY3 in the first 4 bytes.

Figure 2-4 is an example of how you would retrieve all rows whose key starts with PAY3 in bytes 1-4 of the key field.

```
           MOVE 0            TO        ACCTG-COUNT.
           MOVE 4            TO        ACCTG-FG-KEY-LENGTH.
           MOVE 'PAY3'       TO        ACCTG-KEY-AREA.
           MOVE 'FG'         TO        ACCTG-COMMAND.
           PERFORM UNTIL ACCTG-FOUND-CODE EQUAL 'N'
               CALL 'TBLBASE'    USING TB-PARM
                                       ACCTG-COMMAND-AREA
                                       ACCTG-ROW-AREA
                                       ACCTG-KEY-AREA
               IF ACCTG-FOUND-CODE EQUAL 'Y'
      *           selected row retrieved for further use .....
           END-PERFORM.
```

Figure 2-4: Retrieving rows using a partial key

The FG command performs a generic fetch. It will retrieve all rows which match the first portion (PAY3) of the key that you have placed in ACCTG-KEY-AREA. We follow the tableBASE call with a test to determine whether the row exists. Note also that setting a 4 in the ACCTG-FG-KEY-LENGTH field of the command area insures that the first row matching the partial key is retrieved. tableBASE sets various flags in the COMMAND-AREA to keep you informed of the results of your request; the FOUND-CODE is one of the flags set by tableBASE.

Note: In order to ensure consistent operation when searching for matching rows, it is recommended that the COUNT field be set to zero before the first FG operation; thus, if a matching row is found, it will always be the first occurrence.

To obtain a particular row number by position, use the FC (Fetch by Count) command. To search on an entire key, use FK (Fetch by Key) or FN (Fetch Next). Use the Get-series

of commands to cycle through a table: Get First (GF), Get Next (GN), Get Last (GL), and Get Previous (GP).

Updating involves replacing, inserting, and deleting. Updates to a table are based on either a key or a row number. To insert a row based on a key use the IK command. To insert a row based on row number, use the IC (Insert by Count) command. The replace and delete functions operate in a fashion similar to insert. For each function there are two options—key and row number. The commands are RK (Replace by Key), RC (Replace by Count), DK (Delete by Key), and DC (Delete by Count).

Note: In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Store processing

To ensure that any changes made to a table are persistent, you must store the table to a tableBASE library (see "Store Table (ST)" on page 134). When a table is stored, a new generation of the table is created. Unless a table is diverted for storage to another tableBASE library by means of the Table Control command DV (or its associate, DW), a new table generation will always be stored in the library from which it was originally loaded.

Since a new generation is stored before the oldest one is deleted, there must be enough space in the library to temporarily accommodate one additional generation. This means there must be enough space available to hold one more generation of the largest table on the library. When the number of generations required to be kept is exceeded, the oldest generation will be deleted.

Close processing

When you have finished accessing a table, you should Close (CL) it. When a table is closed, the memory occupied by the table definition and table data is released.

The CL command does not automatically store a table. If you wish to store the table, a store command must be issued prior to the close.

Searching libraries

When a table is to be opened for loading into a TSR (local or shared), tableBASE uses a list of libraries (LIB-LIST) given by the application via an ML command (see "Modify Library Search Order (ML)" on page 117). If the application has not done an ML at the time a table is to be opened, a default is assumed by tableBASE. The default is the delivered default library named MAINLIB. When you have more than one tableBASE library, you can fine tune your application's performance by specifying the sequence in which the libraries are searched (see "Modify Library Search Order (ML)" on page 117).

Return values

This section lists the various return codes.

Using the ERROR code

Successful execution of any command sets the ERROR code to zero and returns control to the application program. If tableBASE encounters an error, the ERROR code field of the xxxx-COMMAND-AREA is set appropriately, and if the abend switch is on, the task or application will abend. The details regarding the abend switch can be found with the description of the Change Status command (see "Change Status (CS)" on page 73). In addition Chapter 4 describes a temporary ABEND override.

ERROR subcode

With Version 6, an error subcode has been added. The subcode provides additional information and is meant to be used with the ERROR code to help you pin-point a problem (see "tableBASE error codes" on page 408).

Using the FOUND code

The FOUND code indicates whether a search or retrieval was successful. All commands set the FOUND code in xxxx-COMMAND-AREA to either Y or N. Generally, this flag is set to N, except for the commands listed in Table 2-4 where either setting is valid.

Table 2-1: Commands that set the FOUND code

Purpose	Command
Retrieval	SK, FK, FC, FG, FN, GF, GL, GN, GP
Updating	DK, IK, RK,
	DC (always Yes, if no error),
	IC (always No),
	RC (always Yes, if no error)
Table Control	GD
Library Maintenance	NX

Most common programming errors can be avoided by testing the FOUND code when using these commands.

Note: If you have used date-sensitive processing with earlier releases of tableBASE, you should be aware that those earlier releases allowed a value of X to be placed in the FOUND-CODE when using a Fetch by Key (FK) operation. See Appendix B on page 383 for further details.

Advanced tableBASE functionality

This section details the advanced capabilities of the tableBASE software.

Date-sensitive processing

Occasionally you may have a need to implement different business rules based on a date. tableBASE provides for date-sensitive processing when the following rules are followed:

1. The table must be defined and organized as ascending.

2. Date-sensitive processing can only be used with retrieval commands: FC, FK, FG, FN, GF, GN, GP and GL.

3. To indicate that you wish to process rows based on tableBASE's Date-Sensitive Processing feature, use the FUNCTION-ID field in the COMMAND-AREA. For more information see "13. FUNCTION-ID (halfword binary)" on page 147.

4. Each row in the table must contain both an effective date (specifying when the rule is to become active) and an expiration date (specifying the last day the rule is active). Both these date fields must be in the format YYYYMMDD.

   The effective date is the last 8 bytes of the key. The expiration date is the first 8 bytes following the effective date.

5. The key length of the table must include the length of the effective date field.

6. If you wish to specify the date against which the effective and expiration date will be checked (baseline date), you can do so using the DATE field in the COMMAND-AREA. The default is the system's current date. For more information see "14. FUNCTION-AREA: DATE (8 bytes)" on page 148.

Note: For backward compatibility, the DATERTNX switch controls the value placed in the Found Code for a Fetch by Count operation on a date-sensitive table. For more information see "Date-Sensitive Processing Found Code" on page 244.

When the FUNCTION-ID field in the COMMAND-AREA indicates that Date-Sensitive Processing is operational and a retrieval command is executed that uses a key (FK, FG, FN), the search key is created by using the key specified and the DATE field in the COMMAND-AREA.

In commands that do not require a key to retrieve the row, such as Fetch by Count (FC), or the various Get commands (GF,GN,GP,GL), no key is used and the row returned is based on the command used and the subset of date-applicable rows in the table. For example, a GN used in conjunction with Date-Sensitive Processing will return the next row for which the DATE field in the COMMAND-AREA is greater than or equal to the effective date and less than or equal to the expiration date.

Note: The default date used by Date-Sensitive Processing rolls over at midnight.

Figure 2-5 below displays some sample rows that illustrate the arrangement of the keys required for date-sensitive processing, as described above.

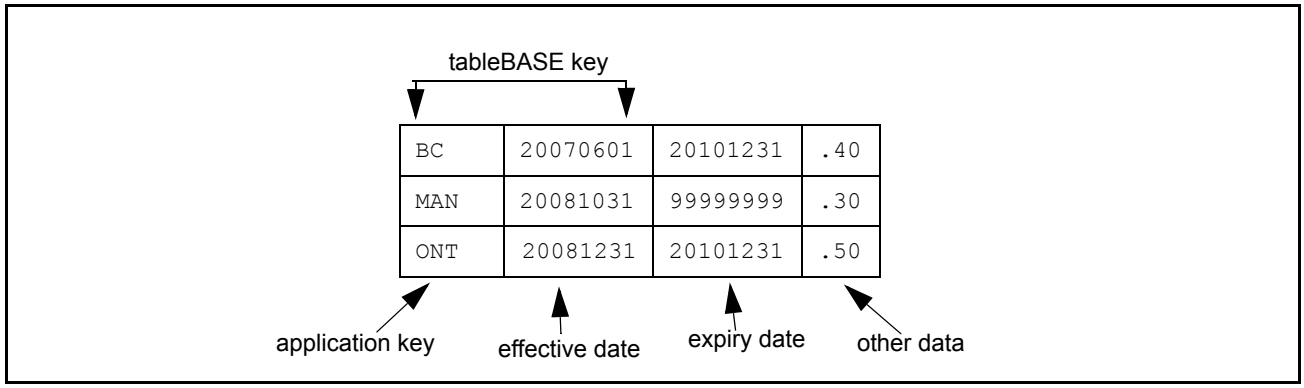

Figure 2-5: Keys required for date-sensitive processing

Generations

You may maintain up to nine generations of a table. When you create a table, you specify how many generations of the table you wish to keep. As you create a new version of a table, tableBASE stores it. When the maximum number of generations has been created, tableBASE automatically deletes the oldest generation when you next create a version.

When you open a table, you may request a specific generation or you may default to the current generation (the generation most recently created). The generation to be opened is supplied as an optional parameter in the call to the tableBASE API. See "GENERATION (fullword binary)" on page 157 for more information.

The absolute generation number of the table being accessed is returned to the user's extended command area, and can be used to verify that the same generation of the table is being used from one call to the next. See "16. RETURNED-ABS-GEN-NO (halfword binary)" on page 148 for more information.

Operational parameters and switches

tableBASE is a very flexible product. You can tailor it to perform optimally in your environment by setting certain parameters and switches for both batch and online environments. These parameters and switches are set at installation time but many of them may be overridden when your program is executed. You can override these parameters and switches either via a file defined in the application's JCL (see "Other parameters" on page 244) or particular tableBASE commands (such as Change Status).

As we discuss each command in the following pages, we will describe those situations where parameters and switches affect the command's operation and tell you how to adjust the parameter and/or switch to meet your needs.

Version control

tableBASE can maintain up to nine generations (or versions) of your tables, so that if the latest generation is wrong, you can quickly revert to an earlier generation. With tableBASE's flexible generation control, you can access any generation you like.

Multitasking batch

tableBASE supports full multitasking in all environments, including batch. Thus, it can be considered a transaction processor in both batch and online environments.

Sophisticated memory management

tableBASE handles tables as data structures resident in main memory (within data spaces). It offers a memory-management system that is optimized for table structures resulting in high performance and easy maintenance.

tableBASE integrates memory management capabilities lacking in many programming languages. It provides:

- Automatic table load/unload facilities
- Efficient main-memory data organizations
- A variety of high performance search methods
- Dynamic runtime expansion of allocated table space
- Dynamic runtime reorganization of tables
- High performance Index structures
- Dynamic runtime Index creation and modification
- Dynamic runtime Alternate Indexes.

Summary of tableBASE commands

The following is a summary of the commands and parameters available when using the tableBASE API, TBLBASE. Chapters 3 and 4 describe the commands and parameters in detail. The parameters listed in Tables 2-2 through 2-6 are in addition to the TB-PARM and COMMAND-AREA parameters. Commands are categorized into the following groups:

- Retrieval commands (Table 2-2 on page 52).
- Update commands (Table 2-3 on page 52).
- Table control commands (Table 2-4 on page 53).
- Library maintenance commands (Table 2-5 on page 54).
- System control commands (Table 2-6 on page 54).

A typical call to tableBASE may look like:

```
CALL   'TBLBASE' USING  TB-PARM
                        COMMAND-AREA
                        PARAMETER-3
                        PARAMETER-4
```

The first parameter, TB-PARM, is optional for all commands; the second parameter, COMMAND-AREA, is mandatory for all commands. tableBASE can determine whether the first parameter specified is the TB-PARM communications area or the COMMAND-AREA. The third parameter must be present if the fourth is used. This is indicated by the notation:

[PARAMETER-3 [PARAMETER-4]]

If KEY-AREA is omitted (commands FK, DK, etc.), the table key is extracted from the ROW-AREA. If other parameters are omitted, default values are used.

Although the TB-PARM parameter is optional, using TB-PARM is highly recommended as it provides significant performance benefits.

Note: If you are using tableBASE VTS (Virtual Table Share), via the VS command or the TB-PARM, these commands will operate in the VTS-TSR. Otherwise, these commands will operate in the local TSR. Generally, library maintenance and system control commands do not operate in the TSR (BN, CS, LS, ML, LL), with the exception of LD, which ceates a temporary table in the current TSR or VTS-TSR.

Table 2-2: Retrieval commands

Command	Description	Parameters		Open status
SK	Search for Matching Key	KEY-AREA	---	Implicit Open 1
FK	Fetch by Key	ROW-AREA	[KEY-AREA]	Implicit Open 1
FN	Fetch Next by Key	ROW-AREA	[KEY-AREA]	Implicit Open 1
FG	Fetch Generic	ROW-AREA	[KEY-AREA]	Implicit Open 1
FC	Fetch by Count	ROW-AREA	---	Implicit Open 1
GF	Get First	ROW-AREA	---	Implicit Open 1
GL	Get Last	ROW-AREA	---	Implicit Open 1
GN	Get Next	ROW-AREA	---	Implicit Open 1
GP	Get Previous	ROW-AREA	---	Implicit Open 1

Table 2-3: Update commands

Command	Description	Parameters		Open status
DK	Delete by Key if found	ROW-AREA	[KEY-AREA]	Implicit Open 2
IK	Insert by Key if not found	ROW-AREA	[KEY-AREA][1]	Implicit Open 2
RK	Replace by Key if found	ROW-AREA	[KEY-AREA][1]	Implicit Open2
DC	Delete by Count	ROW-AREA	---	Implicit Open 2
IC	Insert by Count	ROW-AREA	---	Implicit Open 2
RC	Replace by Count	ROW-AREA	---	Implicit Open 2
MT	EMpty Table		---	Implicit Open 2

Note 1.The KEY-AREA parameter is included for backward compatibility. The table key is extracted from the ROW-AREA. If KEY-AREA is supplied, it is ignored.

Table 2-4: Table control commands

Command	Description	Parameters		Open status
OR	Open for Read	[PASSWORD [GENERATION]]		Any Time
OW	Open for Write	[PASSWORD [GENERATION]]		Not Open, Open for Read
CL	Close Table	---	---	Open
RL	ReLease table	---	---	Open for Write
RF	ReFresh table	---	---	Open for Read
ST	STore table	---	---	Open for Write
GD	Get table Definition	DT-BLOCK	[GENERATION]	Any Time
DD	Dump Definition	truncated DT-BLOCK	[GENERATION]	Any Time
DT	Define Table	DT-BLOCK	---	Not Open
CD	Change table Definition	DT-BLOCK	---	Implicit Open 2
DV	DiVert table to Another library	DDNAME	---	Open
DW	Divert Table to Another library Where it Exists	DDNAME	---	Open
CN	Change Name in TSR	NEW-TABLE-NAME	---	Implicit Open 1
DU	DUmp table Contents	TBACC-DEF	TABLE-AREA	Implicit Open 1
CA	Create Alternate Index definition	DATA-TABLE-NAME	ALT-DEFINITION	Note 1
IA	Invoke Alternate Index	[DATA-TABLE-NAME [ALT-DEFINITION]]		Open

Table 2-5: Library maintenance commands

Command	Description	Parameters		Open status
DG	Delete Generation	[PASSWORD [GENERATION]]		Not Open
XT	Eliminate Table	[PASSWORD]	---	Not Open
CG	Change Generations to keep	[PASSWORD [NEW-GEN-NO]]		Not Open
RN	ReName table In library	NEW-TABLE-NAME	[PASSWORD]	Not Open
NX	Get NeXt table name	[DDNAME [LIB-SPACE]]		Any Time
LD	List Directory	[DDNAME [DIR-SPEC]]		Any Time
DL	Define new tableBASE Library	DDNAME	[LIB-SPACE]	Any Time

Table 2-6: System control commands

Command	Description	Parameters		Open status
ML	Modify Library search order	LIB LIST		Any Time
LL	List Library search order	LIB LIST		Any Time
CS	Change TBLBASE Status	STATUS-SWITCHES		Any Time
LS	List TBLBASE Status	STATUS-SWITCHES		Any Time
LT	List open Tables	LIST-BLOCK	[TABLE-STATS]	Any Time
LV	List VTS settings	VTS-DATA		Any Time
BN	BaNner retrieval	NAME-AREA	[RELEASE-LEVEL]	Any Time
AL	Allocate Library	DDNAME	LIBRARY-ALLOC	Any Time

Table 2-6: System control commands (Continued)

Command	Description	Parameters		Open status
UL	Un-allocate Library	DDNAME		Any Time
SI	Set Indirect	INDIRECT		Any Time
VS	Set Virtual System	VTS-NAME		Any Time
DE	DisEngage			Any Time
XX	tableBASE termination			Any Time

Open status notes

This section provides notes on the Open status of the above commands.

Implicit Open 1

A closed table is implicitly opened for read with any of these commands: SK, FK, FC, FG, FN, GF, GL, GN, GP, CN, and DU.

Implicit Open 2

A closed table is implicitly opened for read with any of these updating commands in batch when using a local TSR: DK, IK, RK, DC, IC, RC, MT, CD. With multi-tasking and multi-user environments, an OW, DT or CN command must be issued before using the commands.

Open

The table must be opened automatically or explicitly with an OR or OW command before issuing any of these commands: CL, ST, DV, DW, and RL.

Not Open

The table must be not open when issuing any of these commands: DT, DG, XT, CG, RN, IA, and CA.

Any Time

The following commands may be performed at any time: NX, LD, GD, ML, LL, CS, LS, BN, SI, DE, LT, and LV.

Note 1

The CA command defines an alternate index definition and saves it on a tableBASE library.

3 tableBASE commands

The tableBASE commands that are described in this chapter apply to batch and online interfaces. Because of restrictions imposed by some online environments, not all commands or features are available in all environments. The variations or limitations imposed by any particular environment are treated separately in Chapter 6 "Environmental interfaces" on page 229.

Retrieval commands

You can locate and retrieve rows from a table by using the following Search, Fetch, and Get commands:

- Search by Key (SK)
- Fetch by Key (FK)
- Fetch by Count (FC)
- Fetch Generic (FG)
- Fetch Next by Key (FN)
- Get First (GF)
- Get Last (GL)
- Get Next (GN)
- Get Previous (GP)

Following the execution of any retrieval command, the FOUND code in the xxxx-COMMAND-AREA is set to show the result of the access request (see "3. FOUND (1 byte)" on page 144). All retrieval commands set the COUNT field in xxxx-COMMAND-AREA, except the Fetch by Count (FC) command.

If a retrieval request succeeds in finding the desired row in the table, a copy of the row is moved to the xxxx-ROW-AREA parameter. The exception is SK, which does not retrieve a row.

Commands that use a full-key search, like SK and FK, always find the first row of a group of rows having duplicate keys (SK just finds the row; FK finds and returns it).

The number of bytes returned by a successful retrieval request is normally determined by the row size defined in the table definition. This can be overridden for the request by setting the ROW-OVERRIDE-LENGTH value in the xxxx-COMMAND-AREA.

If ROW-OVERRIDE-LENGTH is set to zero, the request returns the number of bytes specified in the table definition. Otherwise, ROW-OVERRIDE-LENGTH determines the number of bytes returned. If ROW-OVERRIDE-LENGTH is greater than zero and less than the defined row size for the table, then a truncated row is returned. If ROW-OVERRIDE-LENGTH is greater than the defined row size, then an entire row is retrieved from the table and the remainder of the row is padded with low values to the right of the row.

On completion of the request, tableBASE sets the ROW-ACTUAL-LENGTH field of the xxxx-COMMAND-AREA to indicate the actual row size.

If the table has not been explicitly opened, and IMPLICIT_OPEN is set to Y, a retrieval command will cause tableBASE to attempt an automatic open of the table. If the open is successful, COUNT is reset to zero for all retrieval commands except FC. It is this value which would be used in the execution of the retrieval command. This could affect the operation of count-dependent commands, like FG, FN, GN, GP, which may have started with specific values in COUNT for the first access to a table.

Note: In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Note: GET commands GF, GL, GN and GP return empty rows in Hash tables if the HASH EMPTIES-RETURNED switch of the STATUS-SWITCHES is set to Y. Your program can check for empty rows by examining the key portion of xxxx-ROW-AREA for low values.

If the HASH-EMPTIES-RETURNED switch is set to N, empty rows are skipped automatically. COUNT reflects the physical location of the retrieved row in the table. The Change Status (CS) command may be used to suppress the return of empty rows, often resulting in more streamlined code. More information about the CS command is found on page 73.

Update commands

The Update commands allow a programmer to modify the contents of a table by utilizing the Insert, Delete, Replace and Empty commands. These commands may be used either by Key or by Count, where Key is a search key and Count is the subscript of a row in the table. The update command group consists of:

- Delete by Key (DK)
- Insert by Key (IK)
- Replace by Key (RK)
- Delete by Count (DC)
- Insert by Count (IC)
- Replace by Count (RC)
- Empty the Table (MT)

Following the execution of any update command, the FOUND code is set to Y if the requested row was found, otherwise it is set to N. Depending on the operation, either Y or N could indicate success. For example IK would be successful if a row with a matching key was not found (see "3. FOUND (1 byte)" on page 144). In addition, all commands that use a keyed-search (DK, IK, RK) set the COUNT field in xxxx-COMMAND-AREA.

All commands that use a keyed-search will always target the first of a group of rows having the same key.

If the table has not been explicitly opened, and IMPLICIT_OPEN is set to Y, an update command in a single-tasking, single-user environment will cause tableBASE to attempt an automatic open of the table. If the open is successful, COUNT is not reset to zero for the DC, IC and RC update commands. It is the retained value of COUNT which would be used in the execution of these commands.

The number of bytes moved into the table by a successful insert or replace request is normally determined by the row size defined in the table definition. This can be overridden for the request by setting the ROW-OVERRIDE-LENGTH value in xxxx-COMMAND-AREA.

If ROW-OVERRIDE-LENGTH is set to zero, the request updates the table using the number of bytes specified in the table definition. Otherwise, ROW-OVERRIDE-LENGTH determines the number of bytes used for the update. If ROW-OVERRIDE-LENGTH is greater than zero and less than the defined row size for the table, then the table update uses the row area padded with low values. If ROW-OVERRIDE-LENGTH is greater than the defined row size, the row area will be truncated for the table update.

When ROW-OVERRIDE-LENGTH is not 0, it must be large enough to contain the table key.

On completion of the request, tableBASE sets the ACTUAL-ROW-LENGTH field of xxxx-COMMAND-AREA to indicate the row size.

Note: In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Table control commands

The Table Control commands affect an entire table, as opposed to affecting individual rows within a table. They allow a programmer to manipulate tables with the Open, Close, and Store commands, for example. Other commands provide for defining new tables and performing other table controlling operations. This group consists of:

- Open for Read (OR)
- Open for Write (OW)
- Close Table (CL)
- Release Table (RL)
- Store Table (ST)
- Define Table (DT)
- Get Table Definition (GD)
- Dump Definition (DD)
- Change Table Definition (CD)
- Divert (Non-existing) Table (DV)
- Divert (Existing) Table (DW)
- Change Name (CN)
- Rename Table (RN)
- Dump Table Contents (DU)
- Create Alternate Index Definition (CA)
- Invoke Alternate Index (IA)

Opening an Alternate Index

Indexes are generated dynamically when a table is opened, based on information in the table definition for a primary Index or, in the Alternate Index definition, for an Alternate Index. In general, an Alternate Index is treated like any other table, although it contains no data of its own. Unless otherwise noted, documentation for table processing also applies to Alternate Indexes.

An Alternate Index is created using the CA or IA commands. An Alternate Index may be opened using the open commands OR or OW, or an IA command. When an Alternate Index is opened with an OR or OW command, the Data Table will also be opened if it is not already open. An IA command will only work if the Data Table is already open for read or open for write.

You may define more than one Alternate Index for a particular Data Table. Any changes made using an Alternate Index are made to the Data Table and will be reflected in all open Alternate Indexes based on that Data Table.

To control the user(s) or transactions that perform updates in a multi-user environment after opening of an Alternate Index for write, place a password in the LOCK-LATCH field when opening for write using either an OW or CN command. Only the update commands that have the same password in the LOCK-LATCH field are authorized to perform the update (see "9. LOCK-LATCH (8 bytes)" on page 146).

Open Indirect

In most cases you will open a table directly — you know and specify the name of the table you want to open. In some cases you might want to open a table indirectly — you do not know the name of the table you want to open but can determine it programmatically.

A typical example of the latter occurs when a different table may be opened depending on current conditions. For example, you may open one table for typical customers and one or more different tables for atypical customers. If so, you would store the table names of each table (typical and atypical customers) in a separate table, the primary table. Then, depending on conditions, you would direct tableBASE to open the appropriate table by having it find the name of the indirect table by looking for a key in the primary table.

You do this in two steps. First, you define the current conditions (a search argument) using the SI command. You then set a flag in the command area and open the table. tableBASE will apply the search argument to the table of table names (the primary table) to find a matching row, which names the actual table to be opened and then open the correct table. tableBASE assumes that the search argument begins in position nine of each row of the primary table.

To be specific, move the value I into the INDIRECT-OPEN field of xxxx-COMMAND-AREA—an Indirect open will be performed on the next OR or OW command issued. The TABLE field of xxxx-COMMAND-AREA is set to the primary table name by the application before the call to open.

After returning from a successful Open Indirect, the TABLE field of xxxx-COMMAND-AREA contains the indirect (secondary) table name and the xxxx-COMMAND-AREA is ready for retrieval or updating commands to be issued against this indirect table. The value of I set by the application in the INDIRECT-OPEN field is not retained, rather it is reset to low-values so any subsequent operation may be performed with the same xxxx-COMMAND-AREA.

For more information concerning Open Indirect, see "Set Indirect (SI)" on page 131, and the example "Access a table indirectly" on page 220.

Library maintenance commands

The library maintenance commands provide facilities to retrieve or alter the information contained in a tableBASE library. The command group consists of:

- Delete Generation (DG)
- Eliminate Table (XT)
- Change Generations (CG)
- Rename Table (RN)
- Get Next Table Name (NX)
- List Directory (LD)
- Define New Library (DL).

System control commands

The System Control commands allow the programmer to control certain tableBASE operations. This group consists of:

- Modify Library Search Order (ML)
- List Library (LL)
- Change Status (CS)
- List Status (LS)
- List Open Tables (LT)
- List VTS Settings (LV)
- Banner Retrieval (BN)
- Disengage (DE)
- Allocate (AL)
- Un-allocate (UL)
- Set Indirect (SI)
- Virtual Subsystem (VS).

tableBASE command descriptions

The commands are listed in alphabetic sequence. Full definitions of all parameters are found in the next chapter. The errors listed for each command represents only some of the unique errors the command may receive. Many errors are common to all commands.

Allocate (AL)

Command title Allocate Library

Description This command dynamically allocates a tableBASE library or any other dataset to the region.

COBOL syntax

```
MOVE 'AL'                  TO xxxx-COMMAND.

CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           DDNAME
                           LIBRARY-ALLOC.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "AL", 2 );
TBLBASE( &tbParm, &tbCommArea, pDDname, &tbLibraryAlloc );
```

Parameters DDNAME—the library name that the application will use to refer to the allocated library.

LIBRARY-ALLOC—the Dataset Name, and the share status of the dataset.

Explanation The dataset must exist and be catalogued or the allocation fails.

This command is valid only for the MVS batch, TSO/ISPF, or IMS interfaces. The commands AL and UL will return an error code if they are used with DDNAMEs that are permanently allocated through JCL or the TSO ALLOCATE command.

Return value None

Notes None

Exceptions None

See also None

Banner Retrieval (BN)

Command title Banner Retrieval

Description This command retrieves the client name and address as well as the tableBASE release level that is currently executing.

COBOL syntax

```
MOVE 'BN'                  TO xxxx-COMMAND.

CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           NAME-AREA
                           [RELEASE-LEVEL].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "BN", 2 );
TBLBASE( &tbParm, &tbCommArea, pNameArea, pReleaseLevel );
```

Parameters NAME-AREA

RELEASE-LEVEL (optional)

Explanation Various components use BN to get Release Level and display it. For example, TBEXEC (DK1TEXEC), the primary tableBASE batch utility, uses the BN command to display this information on the first page of its Audit Report.

Return value None

Notes RELEASE-LEVEL is new with Version 6.

Exceptions None

See also None

Create Alternate Index Definition (CA)

Command title

Create Alternate Index Definition

Description

This command creates and stores the definition of an Alternate Index into a tableBASE library.

COBOL syntax

```
MOVE 'CA'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            DATA-TABLE-NAME
                            ALT-DEFINITION.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "CA", 2 );
TBLBASE( &tbParm, &tbCommArea, pDataTableName,
&tbAltDefinition );
```

Parameters

DATA-TABLE-NAME indicates the name of the Data Table. This information is stored with the definition of the Alternate Index.

ALT-DEFINITION—Definition of the Alternate Index

Explanation

The existence of the Data Table is not checked until the Alternate Index is opened (OR, OW) or invoked (IA).

Any number of Alternates Index definitions may be created against a single Data Table.

Return value

Previous releases returned error code 0083 for non-pointer (true) data tables, which were not supported.

Notes

An Alternate Index can be defined on the library with the Create Alternate (CA) command, and then invoked with an Open command. A temporary Alternate Index can be defined with the Invoke Alternate (IA) command without placing it on the library (see "Invoke Alternate Index (IA)" on page 102).

The table must be closed for this command.

Exceptions

Version 6 allows CA to be issued against a true table as well as a pointer data table.

See also

"Temporary Alternate Index" on page 283—illustrates the use of CA and IA in conjunction with other tableBASE commands.

See "Opening an Alternate Index" on page 60.

Change Table Definition (CD)

Command title Change Table Definition

Description This command changes the internal definition of a table (or opened alternate index) in memory and, if necessary, reorganizes the table in memory to match the new definition.

COBOL syntax

```
MOVE 'CD'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           DEFINITION-BLOCK.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "CD", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDefinition );
```

Parameters DEFINITION-BLOCK Any sub-parameter in the DEFINITION-BLOCK that is not zero, spaces or low values, and not equal to the current value for that sub-parameter, will cause the current value to be changed in the table definition except in the following cases:

- The number of GENERATIONS to be kept sub-parameter. The Directory Maintenance command CG must be used to change the number of GENERATIONS to be kept sub-parameter.
- The clearing of certain fields: read and write PASSWORDs and USER-COMMENTS. Because blank sub-parameters are interpreted to mean no change, the Read and Write PASSWORDs and USER-COMMENTS cannot be removed by entering spaces. Instead, an asterisk (*****) must be entered as the first byte of the PASSWORD or USER-COMMENTS field to remove it.

Explanation None

Return value

Error Code 92—In the event of receiving an Error Code 92 create more space in the VTS-TSR or local TSR (not unique to CD).

Error Code 73—Applying the CD command to a table opened for Read is successful if MULTITASKING=N, but returns Error Code 73 if MULTITASKING=Y.

Error Code 14 - The row size must be from 1 to 32767.

Error Code 15 - The key size must be from 1 to 256.

Error Code 17 - The changes in key specifications will not fit within the row area.

Error Code 56 - The search method specified is incompatible with the organization of the table.

Others may apply.

Notes

If the new row size is smaller than the old one, all rows in the table are truncated on the right. If the new row size is greater than the old one, then all rows in the table are padded on the right with low values.

If the table is subsequently stored, it will be stored with the new definition.

Changes to sub-parameters Organization, Key Size and Key Location cause an index to be re-organized to match the new definition. These are the only changes which can cause the index to be re-organized.

Exceptions

None

See also

None

Change Generations (CG)

Command title Change Generations

Description This command modifies the number of generations of a table to be kept.

COBOL syntax

```
MOVE 'CG'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           PASSWORD
                           NEW-GEN-NO.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "CG", 2 );
TBLBASE( &tbParm, &tbCommArea, pWritePassword,
         nGeneration );
```

Parameters PASSWORD—If the table is protected with a write password then the write password is required for this command.

NEW-GEN-NO—a positive number between 1 and 9.

Explanation If the number of generations present exceeds the new number of generations to be kept, the appropriate number of the oldest generations of the table will be deleted.

Return value None

Notes The table must be closed for this command.

Exceptions None

See also None

Close Table (CL)

Command title

Close Table

Description

This command removes the definition and contents of a table from the TSR.

COBOL syntax

```
MOVE 'CL'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "CL", 2 );
TBLBASE( &tbParm, &tbCommArea );
```

Parameters

None

Explanation

If the table was open for write, the table will also be released so that other applications may open it for write.

When it is not desirable to remove a table open for write from memory, the RL command provides an alternative.

A CL command issued on the Data Table closes all of the Alternate Indexes as well.

A CL command on an Alternate Index closes the Alternate Index and will also close the Data Table if no commands have been issued to the Data Table itself (rather than through an Alternate Index) since the Data Table was opened implicitly.

Return value

Version 5—returns Error Code 2 when attempting to close a table with an invalid name (all spaces or zeroes).

Version 6—returns Error Code 12 when attempting to close a table with an invalid name (all spaces or zeroes).
Error Code 6 is returned if the table is not open.

Notes

The CL command will not store the table. To store the table, an ST command must be issued before the CL command.

Exceptions

None

See also

"Store Table (ST)" on page 134

"Release Table (RL)" on page 129

Change Name (CN)

Command title

Change Name

Description

This command changes the name of the table or Alternate Index in memory—the status of the newly named table or Alternate Index is open for write.

COBOL syntax

```
MOVE 'CN'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           NEW-TABLE-NAME.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "CN", 2 );
TBLBASE( &tbParm, &tbCommArea, pNewTableName );
```

Parameters

NEW-TABLE-NAME—the new name for the table or Alternate Index

Explanation

The CN command operates on a Data Table or Alternate Index. As a name change could have repercussions to applications that share the table, caution is advised in its use.

The CN command has different effects depending on whether the table is open when the command is issued, and in the case of an Alternate Index, whether it is Invoked (see "Invoke Alternate Index (IA)" on page 102). These cases are outlined below.

CN can also be used to create new tables from a template-table.

In a multi-user environment, if you want to control the user(s) or transactions that perform updates after the CN command (which causes the newly named table to be open for write), place a password in the LOCK-LATCH field when using the CN command. Only the update commands that have the same password in the LOCK-LATCH field are authorized to perform the update. See page 146.

Return value

If the Data Table is not open and it has a Read password protecting the table in the library, an attempt to use the CN command on the Data Table or an Alternate Index will return error 0030.

Notes

The CN command does not affect the table name in the library.

The new table name indicated by the CN command must be a valid table name (see "Table names" on page 31).

The source table indicated by the CN command must be open or be available in a library in the tableBASE Library List (LIB-LIST).

If an RL command was applied to a newly created table using the Change Name command (CN), in previous releases, this would result in a return code of 38. In version 6, this returns a zero. A subsequent attempt to change the in-memory table back to open for write (OW) will only work in single user batch where exclusive access is assured. See "Release Table (RL)" on page 129.

Exceptions

The table name may not be changed to a name of a table that is already open.

If you want to use the CN command on a table in a VTS-TSR then the VTSNAME must be in the TB-SUBSYSTEM field of the TB-PARM (see "TB-PARM (64 bytes)" on page 168).

See also

"Open for Read (OR)" on page 121

"Open for Write (OW)" on page 123

Open Data Table

If the Data Table is already open, the Data Table name is changed, and if the table was opened for read its status is changed to open for write. The renamed table is then treated as a new table.

In addition, any Alternate Indexes on this Data Table that are open when the CN is processed will have their status changed to open for write and their Data Table will be changed to the renamed table name. Closed Alternate Indexes are unaffected.

Closed Data Table

If the CN command is processed and the Data Table is not already open, the Data Table is opened and the table is renamed. The renamed table is then treated as a new table and is open for write.

Read passwords on the Data Table will cause the CN command to fail. Write passwords have no affect.

Invoked Alternate Index	If the Alternate Index is already opened by an IA command, then the name of the Alternate Index is changed.
Open Alternate Index that was not invoked	If the Alternate Index is already open but not by an IA command, the CN command creates a new table with the new name. This new table uses a copy of the table data from the Data Table that is open in memory with an Index as defined by the Alternate Index. The status of this table is changed to open for write. This table is no longer an Alternate Index but a Data Table.
Closed Alternate Index and open Data Table	If the CN command is issued on an Alternate Index which is not already open, the Alternate Index is opened for write. The CN command causes the creation of a new table with the new name. This table uses a copy of the table data from the Data Table that is already open in memory with an Index as defined by the Alternate Index. This table is then treated as a new table. This table is no longer an Alternate Index but a Data Table, and its status is open for write.
Closed Alternate Index and closed Data Table	If the CN command is issued on an Alternate Index which is not already open, the Alternate Index is opened for write. The CN command causes the creation of a new table with the new name. This table uses a copy of the table data from the Data Table on the tableBASE library with an Index as defined by the Alternate Index. This table is then treated as a new table. This table is no longer an Alternate Index but a Data Table, and its status is open for write.

Change Status (CS)

Command title

Change Status

Description

This command sets the tableBASE control states for:

- abend processing
- wait processing for tables in use
- empty row processing for hash tables
- implicit open processing
- trace processing

COBOL syntax

```
MOVE 'CS'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           STATUS-SWITCHES.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "CS", 2 );
TBLBASE( &tbParm, &tbCommArea, pStatusSwitches );
```

Parameters

STATUS-SWITCHES contains five one-byte status switches followed by 3 reserved bytes (see "STATUS-SWITCHES (8 bytes)" on page 163). Each status switch must contain Y, N, or a space. A switch containing a space is not changed by CS.

For details on each switch, see below.

Explanation

A successful command takes effect immediately and continues until another CS command is executed. If any of the proposed switch settings are invalid, then the command is unsuccessful and the existing switch settings are not altered.

Any table name in the xxxx-COMMAND-AREA is ignored by this command.

Return value

Error 0010 occurs when the switches are not N, Y, or space.

Notes

The initial settings of the tableBASE status switches are determined at product installation time. These initial settings may be different for each environment: batch, CICS, IMS TM, DB2, etc.

It is also possible for the installation to prevent a program from changing any of these initial settings since the ability to change any of the status switches may be disabled at installation time. Any attempt to change a setting that has been disabled will be ignored.

Switch 1 ABEND (abend processing)

Description Suppression of abend processing is performed to allow application programs to continue processing despite tableBASE errors that can be anticipated, detected in the application program, and dealt with in a general error-handling procedure.

Set to Y tableBASE abend processing is to be performed for tableBASE errors 0001-0099 and 1000-1099.

Set to N Abend processing is not to be performed on tableBASE errors 0001-0099 and 1000-1099.

After suppressing abend processing (ABEND = N), the program should check the ERROR code in xxxx-COMMAND-AREA after each call to tableBASE. A non-zero value indicates a processing error or unexpected condition. tableBASE initializes ERROR to zero on each call so that this need not be done by the program.

Set to blank No change is made to the current setting.

Note The Abend Status can be temporarily overridden on a command-by-command basis by the use of the ABEND-OVERRIDE switch in the command area (see "6. ABEND-OVERRIDE (1 byte)" on page 145).

Switch 2 WAIT (wait processing for tables in use)

Description If a user or application requests that a table be opened for write but that table is already opened for read or write in another region of the computer system, the application has two choices: have tableBASE cause the application to wait until the table becomes free (closed or released by the owning region), or to return an ERROR code that indicates the table is not available to the requesting application.

Set to Y If WAIT is set to Y, the requesting application (or transaction) will wait.

Set to N tableBASE does not wait for opened tables.

If Abend processing is enabled, tableBASE abends with USER 0072. If Abend processing is not enabled, the ERROR is set to 0072.

Set to blank No change is made to the current setting.

Note None

Switch 3	HASH-EMPTIES-RETURNED (empty row processing for hash tables)
Description	With HASH-EMPTIES-RETURNED set to N, program code is generally more streamlined since there is no need for the application to check for empty rows. In addition, programs are less likely to be affected by a change of table organization.
Set to Y	tableBASE returns empty rows.
Set to N	tableBASE suppresses the return of empty rows for tables which have a Hash organization. This processing option is for GF, GL, GN, and GP commands only.
Set to blank	No change is made to the current setting.
Note	None

Switch 4	DEFAULT-OPEN (implicit open processing)
Description	The DEFAULT-OPEN Switch controls whether tableBASE will attempt to open a table for a retrieval or update command, and the table is not already open in the TSR.
Set to Y	Implicit open is allowed.
Set to N	Any attempt to access an unopened table with a retrieval command or an update command results in ERROR 0002. A specific OR or OW command must be issued.
Set to blank	No change is made to the current setting.
Note	See "Implicit table opens" on page 42.

Switch 5	TRACE (trace processing)
Description	The TRACE Switch controls whether tableBASE records the most recent commands.
Set to Y	tableBASE automatically records the last ten commands executed per thread. This is done for diagnostic purposes. **Note:** This switch should not be turned on in production as there are performance consequences in its usage. When a process is not working as intended, this setting may help pinpoint a problem via the examination of tableBASE events.
Set to N	tableBASE does not trace commands.

Set to blank No change is made to the current setting.

Note The command trace can be found in the TBDUMP dataset. See "tableBASE user errors" on page 285. Contact DataKinetics support.

Switch 6-8 FILLER (3 bytes)

Description Reserved for future use

Delete by Count (DC)

Command title

Delete by Count

Description

This command deletes a row from a table and places it in xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'DC'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DC", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters

xxxx-ROW-AREA

Explanation

This command deletes a row whose subscript is specified in the COUNT field of xxxx-COMMAND-AREA from the table identified by the TABLE field of the xxxx-COMMAND-AREA. The deleted row is moved into xxxx-ROW-AREA for verification.

The ROW-OVERRIDE-LENGTH can be used to indicate how many bytes of the row are moved into xxxx-ROW-AREA.

Return value

The FOUND field is Y if there was no error and the row is found and deleted, otherwise it is set to N.

Error 0006 subcode 1 occurs when the COUNT is zero.
Error 0006 subcode 2 occurs when the COUNT exceeds the number of rows.

Notes

DC should be used only when the subscript of the row to be deleted is known, usually through a previous successful FETCH or GET. It may not be used to delete empty Hash table entries.

If Delete by Count is used inappropriately, unintended rows may get deleted causing unpredictable results. If you cannot ensure exclusive access to the table, via a LOCK-LATCH password in a multi-user environment such as CICS, it is preferable to use a Delete by Key command.

In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Exceptions None

See also "3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Dump Definition (DD)

Command title

Dump Table Definition

Description

This command is obsolete, but is still supported for compatibility with existing programs. It is superseded by the GD (Get Table Definition) command. The DD command supplies information describing the specified table, but excluding the read and write passwords.

COBOL syntax

```
MOVE 'DD'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           DEFINITION-BLOCK
                           [GENERATION].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DD", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDefinition,
         nGeneration );
```

Parameters

DEFINITION-BLOCK

Explanation

Only the first 64 bytes of the DEFINITION-BLOCK are returned, through field XXXX-ABS-GEN-NO.

Return value

The read and write Password fields will be set to blanks if the table has no passwords, or will be set to Xs if the password has been defined for the table. The information is returned in the DEFINITION-BLOCK parameter.

If the table is found, FOUND is set to Y, otherwise it is set to N.

If the table is already open, the definition for the open generation is returned.

If the table is not open, the current generation is taken from the first tableBASE library where the table is found as specified in the LIB-LIST.

The command shows the time and data of creation for a temporary table (created by DT or IA). Previous releases showed these values only for tables that were stored on a library.

Notes

For an Alternate Index that is not open, row size is returned as zero.

Exceptions

None

See also

"3. FOUND (1 byte)" on page 144

Disengage (DE)

Command title Disengage

Description This command closes any open tableBASE libraries.

COBOL syntax

```
MOVE 'DE'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DE", 2 );
TBLBASE( &tbParm, &tbCommArea);
```

Parameters None

Explanation This command may allow other applications to be started (especially in TSO) that require the allocation of open tableBASE libraries. In order to use DE for switching library ddnames in any interface, there should be no intervening tableBASE command that accesses a library right after the DE command and before the next UL and AL command. In CICS, the AL and UL command are unavailable, but the DE command is, so the same rule applies if somehow it is being used before doing a CICS command to switch library DDNames.

Return value None

Notes None

Exceptions None

See also None

Delete Generation (DG)

Command title Delete Generation

Description This command deletes one generation of a table on the library.

COBOL syntax

```
MOVE 'DG'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           [PASSWORD
                           [GENERATION]].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DG", 2 );
TBLBASE( &tbParm, &tbCommArea, pWritePassword,
         nGeneration );
```

Parameters PASSWORD—If the table has been defined with a write password, then it must be specified as the third parameter of this command.

GENERATION—If the GENERATION parameter is omitted, the most current generation is deleted.

Explanation The generation is deleted from the first of the concatenated libraries in which the table is found. For more information on "concatenated libraries" see "Modify Library Search Order (ML)" on page 117.

Return value Error 0008 occurs when the GENERATION number specified for this table is invalid.

Notes If the generation number is to be specified for a table that has no write password, then a dummy third argument must be provided as a placeholder in the parameter list.

Exceptions None

See also None

Delete by Key (DK)

Command title

Delete by Key

Description

This command deletes a row from a table and places it in xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'DK'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA
                            [xxxx-KEY-AREA].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DK", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea, pKeyArea );
```

Parameters

xxxx-ROW-AREA

xxxx-KEY-AREA — If xxxx-KEY-AREA is omitted, the table key is extracted from xxxx-ROW-AREA.

Explanation

DK searches the table for a row with a key matching the one specified; if found, the row is deleted. The deleted row is moved into xxxx-ROW-AREA for verification.

The ROW-OVERRIDE-LENGTH can be used to indicate how many bytes of the row are moved into xxxx-ROW-AREA.

Return value

If the row is found, COUNT is set to the subscript of the deleted row and FOUND is set to Y.

If the requested row is not found in the table, COUNT will contain the appropriate subscript for a potential insertion and FOUND is set to N.

Notes

None

Exceptions

None

See also

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Define New Library (DL)

Command title

Define New Library

Description

This command initializes the dataset referred to in the DDNAME parameter to be a tableBASE library.

COBOL syntax

```
MOVE 'DL'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           DDNAME
                           [LIB-SPACE].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DL", 2 );
TBLBASE( &tbParm, &tbCommArea, pDDName, &tbLibSpace );
```

Parameters

DDNAME

LIB-SPACE (optional)—Upon return, the LIB-SPACE parameter contains the number of blocks defined for the library and the number of blocks remaining after tableBASE has used some blocks to create the library directory.

Explanation

The dataset must have a disposition of NEW or OLD in the JCL, and have at least 10 blocks allocated.

Return value

Error 0061 subcode 3 occurs when the DL command failed because the library is already open.

Error 0061 subcode 4 occurs when the DL command failed because the DISP must be NEW or OLD.

Error 0061 subcode 5 occurs when the DL command failed because the VSAM library was not defined as reusable.

The error codes listed above are unique to DL; other, more generic errors are possible for DL (40, 61, etc.).

Notes

This command is only available in a single-task batch environment.

Exceptions

None

See also

The chapter "Using TBEXEC (DK1TEXEC)" in the *tableBASE Batch Utilities Guide*—for information on defining and formatting tableBASE libraries

The EXPAND Library command in the *tableBASE Batch Utilities Guide*—for information on library expansion

Define Table (DT)

Command title Define Table

Description DT builds a table in the TSR from the values supplied in the DEFINITION-BLOCK parameter.

COBOL syntax

```
MOVE 'DT'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            DEFINITION-BLOCK.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DT", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDefinition );
```

Parameters DEFINITION-BLOCK

Explanation Following successful execution of this command, the table is open for write.

The table to be defined may be given any valid table name that does not duplicate a table already open in the TSR (see "Table names" on page 31).

In a multi-user environment, if you want to control the user(s) or transactions that perform updates after the DT command (which causes the newly defined table to be open for write), place a password in the LOCK-LATCH field when using the DT command. Only the update commands that have the same password in the LOCK-LATCH field are authorized to perform the update (see "9. LOCK-LATCH (8 bytes)" on page 146).

Should the table eventually be stored, it will be stored to the first library in the ML list at the time of the DT command. Thus, it may be necessary to do an ML command to set the LIB-LIST appropriately before doing the DT command. If the LIB-LIST is empty at the time of the DT, a suitable library for storing the table can be specified by a DV or DW command before any subsequent ST. If an ML command was never issued, this library is assumed to be the default library (the installation default library is MAINLIB).

Return value

Error 0003 occurs when a table with the same name is open in the TSR.

Other possible ERRORs 0012, 0014, 0015, 0016, 0017, 0021, 0041, 0043, 0044, 0051, 0055, 0056, and 0064 are related to invalid sub-parameters.

See "tableBASE error codes" on page 139.

Notes

In previous releases, if an RL command were applied to a newly created table using a Define Table command (DT), this would result in a return code of 38. In version 6, this returns a zero. Subsequent updates to the in-memory table, which is now open for read, will only work in a single user batch environment where exclusive access is assured. Open for read tables cannot be stored to a library. See "Release Table (RL)" on page 129.

Exceptions

None

See also

"Divert (Existing) Table (DW)" on page 88—a DW command is needed before the store if the library contains a table with the same name.

Dump Table Contents (DU)

Command title

Dump Table Contents

Description

This command is used to set up parameters and data to be accessed by the low-level access routine TBACC.

COBOL syntax

```
MOVE 'DU'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            TBACC-DEF
                            TABLE-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DU", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbAccDef, pTableArea );
```

Parameters

TBACC-DEF

TABLE-AREA

Explanation

The contents of the first N rows of the table are moved to TABLE-AREA, where N equals the COUNT value set by the application in xxxx-COMMAND-AREA. The command is normally followed by a call to TBACC. If ERROR 0 is returned, the TBACC-DEF is returned and can be used directly by the TBACC(DK1T0103) subroutine.

Return value

ERROR 0068 is returned if the requested N rows is smaller than the number of rows in the table. The first N rows are returned even when N is less than the number of rows in the table.

Notes

If this command causes an Implicit Open of the table, the count field is not affected.

The DU command only updates the first 29 bytes of the TBACC DEF parameter (see "TBACC-DEF (29 bytes)" on page 167).

Exceptions

None

See also

Chapter 11 "tableBASE subroutines" on page 289—for more information on TBACC

"8. COUNT (fullword binary)" on page 146

Divert (Non-existing) Table (DV)

Command title

Divert Table to a library which does not contain the table

Description

This command causes any subsequent store of the table to be targeted to the tableBASE library specified in the DDNAME parameter—unlike DW, DV is used when the targeted directory does not yet contain a copy of the table.

COBOL syntax

```
MOVE 'DV'                    TO xxxx-COMMAND.
CALL 'TBLBASE' USING         TB-PARM
                             xxxx-COMMAND-AREA
                             DDNAME.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DV", 2 );
TBLBASE( &tbParm, &tbCommArea, pDDName );
```

Parameters

DDNAME

Explanation

This command does not actually store the table; the store must be explicitly performed by issuing an ST command. In batch using a local TSR, if the table was opened with an OR, the DV command promotes the Table to Open for Write and allows the table to be stored. In an online interface, batch multitasking or multitasking environment such as a VTS-TSR, the table must be open for write before issuing the DV.

If the DV is not successful, the DDNAME and DSN will be blank, and the table will be treated like a defined (DT) table.

Return value

Error 0040 subcode 1 occurs if the DDNAME does not exist.

Notes

In prior releases, the DV command did not check whether the target DDNAME was allocated. tableBASE put the target DDNAME in the table definition even if the target DDNAME was not allocated. Subsequent Store commands would fail with return code 40. In Version 6, tableBASE checks to see that the target DDNAME is allocated and returns an error code 40-1 if it is not. The table is then treated like a defined (DT) table.

Exceptions

None

See also

"Divert (Existing) Table (DW)" on page 88

"Store Table (ST)" on page 134

Divert (Existing) Table (DW)

Command title

Divert Table to a library that already contains a table with the same name

Description

This command causes any subsequent store of the table to be targeted to the tableBASE library specified in the DDNAME parameter. Unlike DV, DW is used when the targeted library already contains a table with the same name, in which case an ST command causes the specified table in a TSR to be stored as a new generation of a table in the target tableBASE library.

COBOL syntax

```
MOVE 'DW'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           DDNAME.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "DW", 2 );
TBLBASE( &tbParm, &tbCommArea, pDDName );
```

Parameters

DDNAME

Explanation

If the table on the target library has a write password, then the table being Diverted must have the same write password. If the table on the target library does not have a write password, and the table being Diverted does have a write password, then this write password will be removed.

DW allows a new table (either newly created or new by virtue of a Change Name command) to be stored into a tableBASE library which already has a table by that name.

DW also allows a table to be opened from one library, and stored to another library.

Return value

Error 0040 subcode 1 occurs if the DDNAME does not exist.

Error 0031 occurs if the table being diverted does not contain the identical write password as the target table.

Error 0009 subcode 3 occurs if table does not exist on the target library.

Notes This command does not actually store the table; the store must be explicitly performed by issuing an ST command. If the table was opened with an OR in batch using a local TSR, the DW command promotes the table to Open for write and allows the table to be updated and stored. In an online interface, batch multitasking or multitasking environment such as a VTS-TSR, the table must be open for write.

Exceptions None

See also "Divert (Non-existing) Table (DV)" on page 87

"Store Table (ST)" on page 134

Fetch by Count (FC)

Command title

Fetch by Count

Description

This command fetches a row from a table using the subscript specified in the COUNT field of xxxx-COMMAND-AREA and places it in xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'FC'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "FC", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters

xxxx-ROW-AREA

Explanation

None

Return value

On a successful retrieval, a copy of the row is moved into xxxx-ROW-AREA and the FOUND field is set to Y.

An FC with a negative or zero COUNT, or a COUNT higher than the number of rows in the table, returns with the FOUND field set to N and ERROR code set to zero.

Notes

The HASH-EMPTIES-RETURNED switch setting has no effect on the FC command. Empty rows as well as populated rows are returned to the application. The user program must check for an empty row by examining the key portion of xxxx-ROW-AREA for low values.

FC can be used in conjunction with Date-Sensitive Processing.

Exceptions

None

See also

"Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Fetch Generic (FG)

Command title Fetch Generic

Description The Fetch Generic command retrieves a row whose key matches a partial or generic key provided in the xxxx-KEY-AREA parameter and places it in xxxx-ROW-AREA.

Successive use of the Fetch Generic command allows for the retrieval of all rows matching the partial key.

COBOL syntax

```
MOVE 'FG'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           xxxx-ROW-AREA
                           [xxxx-KEY-AREA].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "FG", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea, pKeyArea );
```

Parameters xxxx-ROW-AREA

xxxx-KEY-AREA (optional: see Warning below)—If the xxxx-KEY-AREA parameter is not present, the generic key is taken from the key location within xxxx-ROW-AREA.

The FG-KEY-LENGTH field of xxxx-COMMAND-AREA is used to indicate the length of the search key. tableBASE will use the delimiter method if the FG-KEY-LENGTH field is zero.

Earlier releases of tableBASE used a delimiter character in the xxxx-KEY-AREA to mark the end of the generic key. This approach continues to be supported in Version 6 for backward compatibility with earlier releases.

The Fetch Generic key delimiter is '*' when distributed with the product, although this may be changed at tableBASE install time. See the Warning below for more details.

If FG-KEY-LENGTH is zero and there is no delimiter in the input xxxx-KEY-AREA, a row's key will have to match the input xxxx-KEY-AREA exactly for the row to be found.

Explanation

The table is examined for a matching key, starting at the row pointed to by the COUNT value. If the row indicated by COUNT matches the generic key, the next row in the table will be returned if it also has a matching partial key. If the COUNT value points to the last matching row, FOUND is set to N (no row is returned).

If the row indicated by COUNT does not match the generic key, the COUNT is reset to zero and a search is made in the table for the first row that matches. If a matching row is found, it is returned. If not, no row is returned.

If COUNT points outside the table, it is set to point to the first row. Please see an Exception to this below.

The COUNT field should be set to zero before the first FG operation; thus, if a matching row is found, it will always be the first occurrence.

Return value

Error 0004 occurs when the table organization is not Sequential.

Repeated use of this command in a programmed loop returns the first and all subsequent rows in the table that have matching keys, until FOUND is returned as N.

FOUND is set to Y if the row is found.

If a row is returned, COUNT is set to the subscript of the returned row. The row is returned in the xxxx-ROW-AREA. If no row is returned, COUNT will contain the appropriate subscript for a potential insertion.

Notes

The FETCH GENERIC command may be used only with tables having a Sequential (ascending or descending) organization.

FG can be used in conjunction with Date-Sensitive Processing.

In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Exceptions

If the delimiter is located in the first position of the input xxxx-KEY-AREA, all rows in the table will match. In this case the command will function like the GN command and the row following the value in the COUNT will be retrieved. If the COUNT points outside the table, FOUND is set to N.

See also

"Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Warning

Although the xxxx-KEY-AREA parameter is optional, it is highly recommended when using a delimiter in the generic key. If xxxx-KEY-AREA is omitted and a match occurs with a partial search key specified in xxxx-ROW-AREA, the retrieved row overlays the search key. Subsequent use of the FG command to search for matching rows, using a search key now lacking the asterisk (*), will return only rows that exactly match the key of the first retrieved row.

The preferred approach is to use the FG-KEY-LENGTH field, avoiding use of a delimiter. If you are using a delimiter and a table key itself contains a delimiter within it, erratic results are possible. Such a situation could occur if the key contains binary values or text that contains the wildcard character. The delimiter may be changed system-wide to some other character. Please see the *tableBASE Installation Guide* for more information on the FGDELIM parameter.

Fetch by Key (FK)

Command title

Fetch by Key

Description

This command fetches a row from a table and places it in xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'FK'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA
                            [xxxx-KEY-AREA].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "FK", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea, pKeyArea );
```

Parameters

xxxx-ROW-AREA

xxxx-KEY-AREA (optional)

Explanation

This command locates a row that has the same key as the one specified in the xxxx-KEY-AREA parameter. If the xxxx-KEY-AREA parameter is omitted, the key is taken from the key location in the xxxx-ROW-AREA parameter.

Return value

If the row is found, FOUND is set to Y, a copy of the row is moved into xxxx-ROW-AREA and COUNT is set to the subscript of the fetched row. If the requested row is not found in the table, FOUND is set to N and the COUNT contains the appropriate subscript for a potential insertion.

Notes

FK can be used in conjunction with Date-Sensitive Processing.

Exceptions

None

See also

"Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Fetch Next by Key (FN)

Command title

Fetch Next by Key

Description

This command fetches a row from a table and places it in xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'FN'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           xxxx-ROW-AREA
                           [xxxx-KEY-AREA].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "FN", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea, pKeyArea );
```

Parameters

xxxx-ROW-AREA

xxxx-KEY-AREA (optional)

Explanation

This command locates a row that is equal to or greater than the key specified in the xxxx-KEY-AREA parameter. If the xxxx-KEY-AREA parameter is omitted, the key is taken from the key location in the xxxx-ROW-AREA parameter.

Return value

The table is searched for the first row having the key that is equal to or greater than the key when the table is in ascending order, and for the first row having a key that is less than or equal to the key when the table is in descending order. FOUND is set to Y, a copy of the row is moved into the xxxx-ROW-AREA and COUNT is set to the subscript of the fetched row. FOUND is set to N only if the end of the table has been reached and key has not been found. In this case the COUNT contains the subscript for one past the end of the table.

Error 0021 is returned if the table organization is not S or D.

Notes

Table organization must be ascending or descending.

FN can be used in conjunction with Date-Sensitive Processing.

To use the FN command to move through a table with ascending organization, increment the key by the smallest unit possible. This will cause FN to find a row that is greater than or equal to the new key which is slightly larger than the key given in the previous FN command. FN will loop through each row in the table regardless of gaps in the key sequence. For tables in descending order, decrement the key.

Exceptions None

See also "Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Get Table Definition (GD)

Command title

Get Table Definition

Description

This command supplies information describing the specified table, including the complete table definition, but excluding the read and write passwords.

COBOL syntax

```
MOVE 'GD'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           DEFINITION-BLOCK
                           [GENERATION].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "GD", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDefinition,
         nGeneration );
```

Parameters

DEFINITION-BLOCK

GENERATION (optional)

Explanation

None

Return value

The read and write Password fields will be set to blanks if the table has no passwords, or will be set to Xs if the password has been defined for the table. The information is returned in the DEFINITION-BLOCK parameter.

If the table is found, FOUND is set to Y, otherwise it is set to N.

If the GENERATION parameter is specified, the definition is obtained from the library even if the table is open. If the GENERATION parameter is not specified then the following occurs:

- if the table is already open, the definition for the open generation is returned
- if the table is not open, the current generation is taken from the first tableBASE library where the table is found as specified in the LIB-LIST.

The command shows the time and data of creation for a temporary table (created by DT or IA). Previous releases showed these values only for tables that were stored on a library.

Notes

For an Alternate Index that is not open, row size is returned as zero.

Exceptions

None

See also

"3. FOUND (1 byte)" on page 144

Get First (GF)

Command title

Get First

Description

This command retrieves the first row from a table and places it in xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'GF'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "GF", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters

xxxx-ROW-AREA

Explanation

For a non-Hash table, COUNT is set to 1. For hash tables, when the HASH_EMPTIES_RETURNED is N, COUNT is set to the subscript of the first occupied row in the table; otherwise, COUNT is set to 1.

Return value

If the table is empty, FOUND is set to N and ERROR code set to zero.

If the command completes successfully, FOUND is set to Y.

Notes

GF can be used in conjunction with Date-Sensitive Processing.

Exceptions

With HASH-EMPTIES-RETURNED set to N, tableBASE suppresses the return of empty rows for tables which have a Hash organization. See "Change Status (CS)" on page 73 for additional information on HASH EMPTIES.

See also

"Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Get Last (GL)

Command title

Get Last

Description

This command gets the last row from a table and places it in xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'GL'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "GL", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters

xxxx-ROW-AREA

Explanation

Since the last subscript is returned in the COUNT field, the GL Command can be used to determine the number of rows in a non-Hash table. The COUNT returned for a GL of a Hash table cannot be used to determine the number of rows in the table.

Return value

If the table is empty, FOUND is set to N and ERROR code set to zero.

If the command completes successfully, FOUND is set to Y.

Notes

GL can be used in conjunction with Date-Sensitive Processing.

Exceptions

With HASH-EMPTIES-RETURNED set to N, tableBASE suppresses the return of empty rows for tables which have a Hash organization. See "Change Status (CS)" on page 73 for additional information on HASH EMPTIES.

See also

"Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Get Next (GN)

Command title

Get Next

Description

This command increments the COUNT field of the xxxx-COMMAND-AREA by 1 and then retrieves the row with that subscript to the xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'GN'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "GN", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters

xxxx-ROW-AREA

Explanation

If the command causes the table to be opened automatically or this is the first command after an explicit open, the input value of COUNT is set to zero by the open. Therefore, if the table is not empty, the first row in the table is retrieved.

Return value

If the table is empty or the resulting COUNT designates a non-existent row, FOUND is set to N and ERROR code set to zero.

If the row is found, FOUND is set to Y.

Notes

GN can be used in conjunction with Date-Sensitive Processing.

In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Exceptions

With HASH-EMPTIES-RETURNED set to N, tableBASE suppresses the return of empty rows for tables which have a Hash organization. See "Change Status (CS)" on page 73 for additional information on HASH EMPTIES.

See also

"Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Get Previous (GP)

Command title

Get Previous

Description

This command decrements the COUNT field of the xxxx-COMMAND-AREA by 1 and then retrieves the row with that subscript to the xxxx-ROW-AREA.

COBOL syntax

```
MOVE 'GP'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "GP", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters

xxxx-ROW-AREA

Explanation

Automatic open of the table causes COUNT to be set to zero; GP will then decrement that, resulting in a non-existent row pointer.

Return value

If the table is empty or the resulting COUNT designates a non-existent row, FOUND is set to N and ERROR code set to zero.

If the row is found, FOUND is set to Y.

Notes

GP can be used in conjunction with Date-Sensitive Processing.

In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Exceptions

With HASH-EMPTIES-RETURNED set to N, tableBASE suppresses the return of empty rows for tables which have a Hash organization. See "Change Status (CS)" on page 73 for additional information on HASH EMPTIES.

See also

"Date-sensitive processing" on page 48

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Invoke Alternate Index (IA)

Command title Invoke Alternate Definition

Description This command opens an Alternate Index table by generating a new Index for the data of an open table.

COBOL syntax

```
MOVE 'IA'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            [DATA-TABLE-NAME
                            [ALT-DEFINITION]].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "IA", 2 );
TBLBASE( &tbParm, &tbCommArea, pDataTableName,
         &tbAltDefinition );
```

Parameters DATA-TABLE-NAME (optional) —the name of the Data Table

ALT-DEFINITION (optional)

Explanation The Invoke Alternate command is primarily used for generating temporary Alternate Indexes. While the IA can also open a permanent Alternate Index, the recommended way to create a permanent Alternate Index is to use the CA command, and then subsequently use the OR or OW commands to open the Alternate Index.

To create a temporary Alternate Index, create the definition for the Alternate Index with the ALT-DEFINITION and DATA-TABLE-NAME parameter. If these parameters are not supplied, the library concatenation list will be searched for an existing Alternate Index based on the table name in the xxxx-COMMAND-AREA.

An IA command will only work if the Data Table is already open for read or write. This command does not work with a linked table. See "Linked tables and TB-PARM" on page 380.

Return value

Error 0080 subcode 1 or subcode 2 may occur because the Data Table is not open or accessible in the TSR.

Error code 0083 is returned if the Data Table is defined with Index=True. Previous versions of Release 6 may have returned 0. Error code 0083 is the correct response.

Other possible ERRORs 0015, 0016, 0017, 0021, 0055, 0056, 0081, and 0087 are related to invalid sub-parameters.

For a list of tableBASE error codes, see "tableBASE error codes" on page 408.

Notes

A close command will close only the Alternate Index. If a close command is issued against the Data Table, all the Alternates are closed as well.

If an ST command is issued against an Alternate Index, the Data Table is stored.

A Data Table may have any number of Alternate Indexes.

Exceptions

None

See also

"Example using Alternate Indexes" on page 283—illustrates the use of CA and IA in conjunction with other tableBASE commands

"Create Alternate Index Definition (CA)" on page 65

"Opening an Alternate Index" on page 60

"Glossary" on page 26—for a definition of the term Data Table

Insert by Count (IC)

Command title Insert by Count

Description This command inserts a new row into a table.

COBOL syntax

```
MOVE 'IC'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "IC", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters xxxx-ROW-AREA—contains the content to be inserted into the new row.

Explanation The content of xxxx-ROW-AREA is inserted into the position within the table identified by the value of COUNT.

No checking of key sequence is performed using this command. Insert by Count should only be used when the subscript of the row to be inserted is known, usually through a previous retrieval.

If the table is sequential or user ordered, all rows with a subscript greater than that of the COUNT field will be moved up in position to make space for the new row to be inserted.

If the table is randomly ordered, an Insert by Count always results in the new row being added to the end of the table. The value in the COUNT field, if valid, is ignored.

Return value If the COUNT field of xxxx-COMMAND-AREA is valid, the insert will be performed.

ERROR 0006 subcodes 1 and 2 indicate that the COUNT specified is out of bounds.

Notes

If Insert by Count is used inappropriately; for instance, against a table with a hash index, rows may get out of sequence causing unpredictable results. If you cannot ensure exclusive access to the table, via a LOCK-LATCH password in a multi-user environment such as CICS, it is preferable to use an Insert by Key command.

In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Exceptions

None

See also

"8. COUNT (fullword binary)" on page 146

Insert by Key (IK)

Command title Insert by Key

Description This command inserts a new row into a table.

COBOL syntax

```
MOVE 'IK'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA
                            [xxxx-KEY-AREA].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "IK", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters xxxx-ROW-AREA

xxxx-KEY-AREA — included for backward compatibility. The table key is extracted from xxxx-ROW-AREA, and the xxxx-KEY-AREA parameter is ignored.

Explanation The IK command searches the table for a row with a key matching the one specified. If it is not found, then the contents of xxxx-ROW-AREA are inserted into the proper position within the table based upon the results of the search and the table organization. For Random and User ordered tables, the new row is inserted at the end. If the key is found in the table, then the new row is not inserted as the key already exists.

Return value If the row is inserted in the table, COUNT contains the subscript for the inserted row and FOUND is set to N. If the row is found, FOUND is set to Y, the row is not inserted in the table and COUNT is set to the subscript of the found row.

Notes None

Exceptions In a very large Random or User Ordered table, populating an entire table with this command could be quite slow, as this organization forces serial searches and these become slower as the table grows. If key uniqueness checking is not required the IC command is the fastest way to populate a Random table (see "Insert by Count (IC)" on page 104).

See also "3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

List Directory (LD)

Command title List Directory

Description This command creates a library directory in a table identified by the TABLE field in xxxx-COMMAND-AREA.

COBOL syntax

```
MOVE 'LD'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           [DDNAME
                           [DIR-SPEC]].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "LD", 2 );
TBLBASE( &tbParm, &tbCommArea, pDDName, &tbDirSpec );
```

Parameters

DDNAME (optional)—the name of the library. If DDNAME is spaces or low values, or the parameter is not provided, then all libraries in the tableBASE Library List (LIB-LIST) are included in the directory.

DIR-SPEC (optional)—specifies which tables are to be selected and which information format is to be used for the directory list.

The TABLE-NAME-MASK field of the DIR-SPEC parameter specifies the tables to be selected. A wild-card character can be used to identify the tables to be selected.

The DIRTYPE field of the DIR-SPEC parameter can be:

- **T** = All generations of all tables (default)
- **V** = View tables (based on tablesONLINE definition); all table names beginning with x'80' to x'BF', including lower-case characters
- **D** = Latest generation of the Data Tables; all table names beginning with x'C0' to x'FF', including lower-case characters, including all upper-case characters and numbers. tablesONLINE Views are not returned for DIRTYPE=D.

Explanation

DIRTYPE = D. The directory table has a row size of 74 and a key size of 16 beginning at location 1. The format of a data directory entry (row) is:

```
01  DATA-DIRECTORY-ENTRY.
    05  DATA-NAME           PIC X(8).
    05  DATA-DDNAME         PIC X(8).
    05  DATA-ORG            PIC X.
    05  DATA-METHOD         PIC X.
    05  DATA-TYPE           PIC X.
    05  DATA-SMC            PIC X.
    05  DATA-RSZ            PIC S9(9) COMP.
    05  DATA-KSZ            PIC S9(9) COMP.
    05  DATA-KLOC           PIC S9(9) COMP.
    05  DATA-ROWS           PIC S9(9) COMP.
    05  DATA-GENERATIONS    PIC S9(4) COMP.
    05  DATA-DATE-TIME      PIC X(12).
    05  DATA-DATA-TABLE     PIC X(8).
    05  DATA-VIEW-NAME      PIC X(8).
    05  DATA-USERID         PIC X(8).
```

DIRTYPE = V. The directory table has a row size of 44 and a key size of 16 beginning at location 1. The format of a view directory entry (row) is:

```
01  VIEW-DIRECTORY-ENTRY.
    05  VIEWDE-NAME         PIC X(8).
    05  VIEWDE-DDNAME       PIC X(8).
    05  VIEWDE-DATE-TIME    PIC X(12).
    05  VIEWDE-DATA-TABLE   PIC X(8).
    05  VIEWDE-USERID       PIC X(8).
```

DIRTYPE = T. The directory table has a row size of 70 and a key size of 18 beginning at location 1. The format of a total directory entry (row) is:

```
01  TOTAL-DIRECTORY-ENTRY.
    05   TOTAL-NAME         PIC X(8).
    05   TOTAL-DDNAME       PIC X(8).
    05   TOTAL-ABS-GEN-NO   PIC S9(4) COMP.
    05   TOTAL-MAX-GEN-NO   PIC S9(4) COMP.
    05   TOTAL-ORG          PIC X.
    05   TOTAL-METHOD       PIC X.
    05   TOTAL-TYPE         PIC X.
    05   TOTAL-SMC          PIC X.
    05   TOTAL-RSZ          PIC S9(9) COMP.
    05   TOTAL-KSZ          PIC S9(9) COMP.
    05   TOTAL-KLOC         PIC S9(9) COMP.
    05   TOTAL-ROWS         PIC S9(9) COMP.
    05   TOTAL-GENERATIONS  PIC S9(4) COMP.
    05   TOTAL-DATE-TIME    PIC X(12).
    05   TOTAL-DATA-TABLE   PIC X(8).
    05   TOTAL-USERID       PIC X(8).
```


Return value

The COUNT field contains the number of directory entries placed into the table.

Errors 0003 subcodes 2-5 can occur when reusing an existing table.
Error 0050 occurs if you specify an invalid directory type.
Error 0006 can occur if the LD uses an invalid VTS redirect.

Notes

LD does not store the directory list table on any library; if the table is to be stored, an ST command must be issued.

The directory list table is built in the local TSR unless the TBPARM subsystem name field associated with the tableBASE call designates a VTS-TSR.

If TABLE is already open and has the same format as the current request, then the new information will be appended to the end of the table and the COUNT field will be updated to reflect the total number of directory entries in the table.

Exceptions

None

See also

"DIR-SPEC (9 bytes)" on page 157

List Library (LL)

Command title List Library Search Order

Description This command copies the current tableBASE library search list into the LIB-LIST parameter.

COBOL syntax

```
MOVE 'LL'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           LIB-LIST.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "LL", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbListLib );
```

Parameters LIB-LIST

Explanation This capability is for use by routines that must modify the library search order to their own libraries. The LL command lets a routine record the current library search order before the routine modifies the library search list for its own use; the search order can then be restored, using ML, by the routine before it returns to its caller.

Return value None

Notes None

Exceptions None

See also None

List Status (LS)

Command title List Status

Description This command copies the current tableBASE status switch settings to the STATUS-SWITCHES parameter.

COBOL syntax

```
MOVE 'LS'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           STATUS-SWITCHES.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "LS", 2 );
TBLBASE( &tbParm, &tbCommArea, sStatusSwitches );
```

Parameters STATUS-SWITCHES

Explanation This capability is for use by routines that must set the status switches for their own operation. Each such routine should restore the original status (using CS) before it returns to its calling program.

Return value None

Notes None

Exceptions None

See also None

List Open Tables (LT)

Command title List Open Tables

Description The command provides usage statistics related to the TSR and the current tables opened in the TSR to assist in optimization and capacity planning.

COBOL syntax

```
MOVE 'LT'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            LIST-BLOCK
                            [TABLE-STATS].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "LT", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbListBlock,
         &tbTableStats );
```

Parameters LIST-BLOCK

If the fields LIST-FROM and LIST-REQD in the LIST-BLOCK parameter are zero, then nothing will be returned to the TABLE-STATS parameter. In this case, the value returned in the LIST-TOTAL field will show the number of open tables in the TSR and this can then be used to control the amount of space required for the TABLE-STATS parameter in a subsequent execution of another LT command.

The FUNCTION-ID in the command area can be set to 16 to return all output fields in this area - for more details, see Chapter 4 “LIST-BLOCK (88 bytes)” on page 159.

TABLE-STATS (optional)

Explanation

The statistics provided by LT are stored in the LIST-BLOCK parameter and include:

- number of Open tables
- Open Tables size—storage used by all tables currently open.
- Open Tables Size High Water Mark—the largest amount of storage used by all open tables up until now
- TSR High Water Mark—the largest amount of the TSR utilized up until now
- TSR Size Utilized—the amount of TSR currently being used.
- TSR Size—the actual TSR size allocated
- Strobe Interval—the number of tableBASE accesses between each automatically generated snapshot of the TSR statistics to the Table Space Report
- The total call count for all tables in the TSR or the TSR Access counter
- The maximum number of tables in the TSR.

Explanation (cont'd)

See "LIST-BLOCK (88 bytes)" on page 159 for a complete description of the LIST-BLOCK parameter.

For each table selected in the TSR the following information is provided in the TABLE-STATS parameter:

- Table Name
- Open Status (open for write or open for read)
- Whether the table is in a local TSR or a VTS-TSR
- Whether Alternate Indexes are invoked against this table (Y or N)
- Number of accesses to the table since it was opened or the Table Access counter
- Table Size
- Number of rows in table
- Number of rows before table is automatically expanded
- Data table name if the table is an Alternate Index or the name of the VTS-TSR if the table is linked to a VTS-TSR
- Number of updates to the table since it was opened or the Table Update counter

See "TABLE-STATS" on page 165 for a complete description of the TABLE-STATS parameter.

The TSR Access, Table Access and Table Update counters are internal counters that are updated by tableBASE each time certain commands are executed successfully; they are reflected in the Strobe Report as well as the LIST-BLOCK and TABLE-STATS parameters used in the LT command. The following is a list of the commands that update each counter.

For the TSR Access counter:
Retrieval commands: FC, FG, FK, FN, GF, GN, GL, GP, SK
Update commands: DC, IC, RC, DK, IK, RK, MT
Table Control commands: OR (including implicit opens), OW (including implicit opens), RL, RF, ST, DT, CD, DV, DW, CN, DU, IA, DD.

For the Table Access counter:
Retrieval commands: FC, FG, FK, FN, GF, GN, GL, GP, SK
Update commands: DC, IC, RC, DK, IK, RK, MT
Table Control commands: OR (including implicit opens), OW (including implicit opens), RL, RF, ST, DT, CD, DV, DW, CN, DU, IA, DD.

For the Table Update counter:
Update commands: DC, IC, RC, DK, IK, RK, MT
Table Control commands: RL, ST, DT, CD, DV, DW.

Return value None

Notes This command is not usually coded in a program. Support for it has been incorporated in the command processors TBDRIVER (DK1TDRV) and TBDRIVC (DK1TDRVC). For information on using TBDRIVER and TBDRIVC, see Chapter 8 on page 245 and Chapter 9 on page 273.

Using these development tools, the LT command produces formatted output of the current state of the TSR (see Figure 3-1 below). This command might be used in utility transactions. A transaction that closes all open tables is a good example. It could do an LT to see the TSR information just prior to closing all tables.

Exceptions None

See also Chapter 8 “tableBASE driver command processor for CICS” on page 245

Chapter 9 “TBDRIVER command processor for Batch/TSO” on page 273.

Note: In the Figure below, **ACCESS COUNTS** shows the value in the TSR Access counter; **CALLS** shows the value in the Table Access counter, and **UPDATES** shows the value in the Table Update counter.

```
*** ENTER COMMAND (OR /* TO END) ***
LT
                  ------- HIGH WATER MARKS ----- --- CURRENT SPACE ---        --- ACCESS COUNTS ----
      TSR           OPEN         TSR          TSR      OPEN         TSR               TSR
    ENTRIES        TABLES       USAGE         SIZE     TABLES       USAGE             COUNT
          1           12K         44K      51,200K        12K         44K                 0
-------------------------------------------------------------------------------------------------------------

REF   NAME     W/R L/V IA?      CALLS      SPACE       ROWS RWS-BF-EXP BASE/VTS       UPDATES  DATE-TIME
00001 EXAMPLE  R   L   N  0000000001 0000012288 0000000029 0000000065             0000000000 200902271401
  ************* END OF TABLES ***************

*** ENTER COMMAND (OR /* TO END) ***

DRIVER - tableBASE V603  - MAX ERRLVL 00000
```

Figure 3-1: Sample LT command output from TBDRIVER

List VTS Settings (LV)

Command title

List VTS Settings

Description

The command provides VTS-TSR name and parameter information from tableBASE.

COBOL syntax

```
MOVE 'LV'                   TO xxxx-COMMAND
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            VTS-DATA
```

C syntax

```
memcpy( tbCommArea.tbCommand, "LV", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbVtsData);
```

Parameters

VTS-DATA

Explanation

The TBOPT VTS parameter settings are stored in the VTS-DATA parameter and include:

- VTSFIRST—the name of the VTS-TSR that is to be searched for tables before searching any libraries listed in the tableBASE library-list
- VTSLAST—the name of the VTS-TSR that is to be searched for tables after searching any libraries listed in the tableBASE library-list
- VTSNAME—the name of the VTS-TSR region to be accessed by TBCALLV and TBASEV interfaces
- TPVMNAME—the name of the TPVM (VTS Manager) that is in effect from TBOPT settings for tableBASE Process Manager processing. If this does not show, processing is taking place under non-tableBASE Process Manager operation, or under the TPVM *compat*.
- VTSPREFIX—the prefix characters used to designate a VTS, when a VTS is to be included in the ML search order list

Return value

None

Notes

This command is only available from user-written applications. It is not supported in DK1TDRV (TBDRIVER) or DK1TDRVC (TBDRIVC).

Exceptions

None

See also

For information about VTS parameters in TBOPT, see Appendix C on page 389.

Modify Library Search Order (ML)

Command title Modify Library Search Order

Description The ML command sets or changes the order in which tableBASE libraries are searched.

COBOL syntax

```
MOVE 'ML'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           LIB-LIST.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "ML", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbLibList );
```

Parameters LIB-LIST

Explanation ML causes tableBASE to set a library search list by logically concatenating libraries whose DDNAMEs are in LIB-LIST. This permits an automatic search through up to 10 libraries for a table when it is to be opened. The search sequence is specified by the order of the DDNAMEs in the LIB-LIST parameter. The end of the list is indicated by an entry consisting of blanks or low values.

During any program execution, until an ML command is issued to modify the library list, the library used for all tableBASE processing defaults to MAINLIB unless it is overridden with the TBOPT parameters LIB01, LIB02, etc., or by installation defaults.

The ML command can be used repeatedly to alter the order of library concatenation during the execution of a program to affect subsequent operations. The command generates no I/O operations.

A VTS-TSR may also be included in the search list by specifying the name of the VTS-TSR prefixed by "VTS:" in LIB-LIST, for example, VTS:DKL1, instead of a library DDNAME. If a table to be opened already resides in the VTS-TSR, the table will be linked directly to the open one in the VTS-TSR, and will not be opened in the local TSR. A linked table is read-only. If the table is to be opened into a VTS-TSR (e.g. using the VS command or TB-SUBSYSTEM field of the TB-PARM parameter), then any VTS-TSRs in the search list will be bypassed.

Return value None

Notes

The primary purpose of the ML command is to control the source libraries where tables may be found at open time, not the target libraries where they are stored. Tables are normally stored in the same library from which they were originally opened. The only impact the ML command has on storing tables relates to storing a new table. A new table will be stored on the first library in the concatenated list, determined at the time the table is defined.

Exceptions

None

See also

"Divert (Non-existing) Table (DV)" on page 87 and "Divert (Existing) Table (DW)" on page 88—for information on how to alter the target library into which a table is to be stored

"Define Table (DT)" on page 84.

Empty the Table (MT)

Command title Empty the Table

Description This command removes all of the rows from the copy of the table that is open in the TSR.

COBOL syntax

```
MOVE 'MT'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "MT", 2 );
TBLBASE( &tbParm, &tbCommArea );
```

Parameters None

Explanation None

Return value None

Notes The table remains open, with free space allocated, ready to be re-populated with new data.

Exceptions None

See also None

Get Next Table Name (NX)

Command title

Get Next Table Name

Description

This command places the name of the next table in the library directory into the TABLE field of xxxx-COMMAND-AREA.

COBOL syntax

```
MOVE 'NX'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            [DDNAME
                            [LIB-SPACE]].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "NX", 2 );
TBLBASE( &tbParm, &tbCommArea, pDDName, &tbLibSpace );
```

Parameters

DDNAME (optional)—the library to be examined. If the DDNAME parameter is not provided, the first library of the current library search list is used.

LIB-SPACE (optional)

Explanation

The setting of the TABLE field before the NX is executed identifies the last table retrieved, and tableBASE fetches the next table name from the directory. Library directory entries are stored in EBCDIC collating sequence. To retrieve the first table name in the directory, set TABLE to low values or blanks.

Return value

If no table in the directory exists with a name greater than the input TABLE value, FOUND is set to N, otherwise it is set to Y.

If the LIB-SPACE parameter is supplied, the number of blocks originally defined for the library and the number of blocks remaining will be returned.

ERROR 0040 subcode 1 occurs when the DDNAME does not exist.

Notes

None

Exceptions

None

See also

"3. FOUND (1 byte)" on page 144

Open for Read (OR)

Command title

Open for Read

Description

This command loads into the specified TSR a copy of a specified generation of a specified table—the status of the table is opened for read.

COBOL syntax

```
MOVE 'OR'                TO xxxx-COMMAND.
CALL 'TBLBASE' USING     TB-PARM
                         xxxx-COMMAND-AREA
                         [PASSWORD
                         [GENERATION]].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "OR", 2 );
TBLBASE( &tbParm, &tbCommArea, pReadPassword,
         nGeneration );
```

Parameters

PASSWORD (optional)—if the table has been defined with a read password then the correct read password must be supplied in the PASSWORD parameter. A valid write password is also acceptable.

GENERATION (optional)—the GENERATION parameter may be specified as a relative generation (negative number) or as an absolute generation (positive) number; zero may be used to designate the current generation. If GENERATION is given, and the table has no password, a dummy PASSWORD parameter must be specified as a placeholder in the parameter list.

Explanation

The table to be opened is located by searching through a list of libraries, the LIB-LIST. When a table is found on a library, the table definition is retrieved, an appropriate area in the TSR is allocated, and the table is loaded into that area.

Return value

Error 0009 occurs when the table is not found.

Error 0030 occurs when an incorrect password is used.

Error 0033 occurs when a later generation exists in the library than the one already loaded in the TSR.

Error 0086 occurs when an open is attempted on an Alternate Index when a Data Table is already opened via a link to a VTS-TSR.

Notes

A read password only protects the document from being opened for read or write, it does not provide read protection when it is open in the TSR. Others using the tableBASE API can also view the table contents.

Exceptions

An OR issued for an absolute generation of a table already open for read or write has no affect. The data is not reloaded.

If a relative generation is used to open a table, and subsequently another OR is aimed at the same relative generation, problems can arise. New generations may be created by other tasks, transactions and/or regions, changing the relative number, and therefore pointing to an incorrect generation of the table. ERROR code 0033 is returned to indicate this problem to the user.

If the table to be opened already resides in a VTS-TSR that is used from the LIB-LIST, the table will be linked to the copy in the VTS-TSR. No copy will be created in the local TSR.

If a table is opened, either by an implicit or explicit OR, in a DB2 region or multi-task batch region, or a CICS TS or IMS TM region for shared access by multiple users, users will be permitted to issue read-only commands or the CN command. In a single-task batch region, a table opened for read may also be updated, but it cannot be stored back in the library unless a subsequent OW is done.

See also

"Modify Library Search Order (ML)" on page 117—for information on the library search list

"Set Indirect (SI)" on page 131

"Create Alternate Index Definition (CA)" on page 65

"Invoke Alternate Index (IA)" on page 102

"Opening an Alternate Index" on page 60

"Implicit table opens" on page 42

Open for Write (OW)

Command title

Open for Write

Description

This command loads into the specified TSR a copy of a specified generation of a specified table—the status of the table is opened for write.

COBOL syntax

```
MOVE 'OW'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            [PASSWORD
                            [GENERATION]].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "OW", 2 );
TBLBASE( &tbParm, &tbCommArea, pWritePassword,
         nGeneration );
```

Parameters

PASSWORD (optional)—if the table has a write password then the correct write password must be supplied in the PASSWORD parameter. If the table has only a read password defined, then the write password defaults to the read password, and it must be supplied for the OW command to succeed.

GENERATION (optional)—if an earlier generation of the table is required instead of the current generation, then the GENERATION parameter must be supplied. It may be specified as a relative generation (negative number) or an absolute generation (positive number); zero may be used to designate the current generation. If GENERATION is given, and the table has no password, a dummy PASSWORD parameter must be specified as a placeholder in the parameter list.

Explanation

The table is located by searching through a list of libraries. When a table is found in a library, the table definition is retrieved, an appropriate area is allocated in the TSR, and the table is loaded into that area. Subsequent space requirements due to table expansion are satisfied from the TSR and space is limited by the TSR size.

In a multi-user environment, if you want to control the user(s) or transactions that perform updates after the OW command, place a value in the LOCK-LATCH field when using the OW command. Only the update commands, that have the same value in the LOCK-LATCH field are authorized to perform the update (see "9. LOCK-LATCH (8 bytes)" on page 146).

Return value

ERROR 0003 occurs when the table is already open for write.
ERROR 0009 occurs when the table is not found.
ERROR 0030 occurs when an incorrect PASSWORD is used.

ERROR 0033 occurs when a later generation exists on the library than the one already loaded in the TSR.

ERROR 0072 occurs when the table is already open for write in another region.

ERROR 0086 occurs when an open is made on an Alternate Index and the Data Table is already opened via a link to a VTS-TSR.

Notes

1. Only tables that have an open for write status can be stored. The OW command prevents other regions from opening the same table for write. Once the updates are made to the table and the table is stored, issuing a CL or RL command releases update control of the table so that other applications/users can open the table for write.

2. An Alternate Index may be opened using an OW command. If any Alternate Index is opened for write, then the Data Table and all related Alternate Indexes that are currently open will also be promoted to the status of open for write. See "Opening an Alternate Index" on page 60.

3. If an ST command is subsequently issued against an Alternate Index, the Data Table is stored. The primary definition that is used to build the primary Index of the Data Table is also updated on the library. The changes to the Data Table will be reflected in all open Alternate Indexes.

4. If the table is already open for read in a TSR (local or shared) when an OW command is issued against it, the generation of the table in memory will be verified against the current generation in the library and, if it matches, the status of the table will be promoted to open for write without data being reloaded from the library. This is to ensure that no changes have been made to the table from another region. If a newer version exists in the library, the OW is rejected with ERROR set to 0033.

Exceptions

None

See also

"Modify Library Search Order (ML)" on page 117—for information on the library search list

Replace by Count (RC)

Command title

Replace by Count

Description

This command replaces a row that already exists in a table.

COBOL syntax

```
MOVE 'RC'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "RC", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters

xxxx-ROW-AREA

Explanation

The content of the xxxx-ROW-AREA replaces the indicated row in the table.

Return value

FOUND code is always Y after this operation, unless there is an error.

ERROR 0006 subcodes 1 and 2 indicate that the COUNT specified is out of bounds.

ERROR 0006 subcode 3 indicates you are trying to replace an empty Hash table entry.

Notes

No checking of key sequence is performed using this command. Replace by Count should be used only when the subscript of the row to be replaced is known, usually through a previous successful retrieval.

If Replace by Count is used inappropriately, rows may get out of sequence or you may be replacing unrelated rows, causing unpredictable results. If you cannot insure exclusive access to the table, via a LOCK-LATCH password in a multiuser environment such as CICS, it is preferable to use a Replace by Key command.

In a multi-tasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

Exceptions

RC may not be used to replace empty Hash table entries.

See also

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Refresh Table (RF)

Command title

Refresh Table

Description

The RF command refreshes a Data Table that is opened for read in a TSR. Any Alternate Indexes for that Data Table that are opened for read will automatically be refreshed.

COBOL syntax

```
MOVE 'RF'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "RF", 2 );
TBLBASE( &tbParm, &tbCommArea );
```

Parameters

None

Explanation

The Refresh command will refresh a table that is opened for read and all open Alternate Indexes in a TSR, from the latest generation of the table in the library. The table will always be refreshed even if it the currently open version is the latest version of the table. If the LIB-LIST is changed after the table has been opened, the refreshed version may be loaded from a different library. See "Library search order and uniqueness of table names" on page 36.

To minimize the time in which access to the table is restricted, the affected Data Table and any alternates are reloaded while accesses are continuing with the original open copy in the TSR. Only after the refreshed copy and any alternates are completely loaded, internal pointers are switched to the refreshed copy. Access is restricted for the instant the internal pointers are switched.

The RF command only works with a Data Table, not Alternate Indexes.

Return value

ERROR 0009 subcode 2 occurs when a refresh is attempted on a table that is not open.

ERRORs 0093 subcodes 1-7 occur when tableBASE discovers a difference in attributes of the current data table, or opened alternate of the table, with the refreshed version.

ERROR 0085 occurs if the RF command is attempted against an Alternate Index.

For the list of tableBASE error codes, see "tableBASE error codes" on page 408.

Notes

The RF command can also be invoked from the batch TBDRIVER (see "TBDRIVER (DK1TDRV) commands" on page 275). The batch TBDRIVER allows for the use of wildcards (see Table 9-2, Note 2–Note 5 on page 282).

Exceptions

Only tables opened for read may be refreshed. If a table being refreshed is very large, there may not be adequate space in the TSR, as both the current version and the refresh version are resident in the TSR simultaneously. The maximum size table that can be refreshed is 1G, and only when the TSR size is 2G.

See also

None

Replace by Key (RK)

Command title Replace by Key

Description This command replaces a row that already exists in a table.

COBOL syntax

```
MOVE 'RK'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            xxxx-ROW-AREA
                            [xxxx-KEY-AREA].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "RK", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea );
```

Parameters xxxx-ROW-AREA

xxxx-KEY-AREA — included for backward compatibility. The table key is extracted from xxxx-ROW-AREA, and the xxxx-KEY-AREA parameter is ignored.

Explanation It searches the table for a row with a key matching the one specified. The key is obtained from the xxxx-ROW-AREA. The xxxx-KEY-AREA parameter is not used. If it is supplied in the call, it is ignored.

Return value If the row is found, FOUND is set to Y, the content of the xxxx-ROW-AREA replaces that row in the table and COUNT is set to the subscript of the replaced row.

If the requested row is not found in the table, FOUND is set to N, the row is not replaced and COUNT will contain the appropriate subscript for a potential insertion.

Notes None

Exceptions None

See also "3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Release Table (RL)

Command title

Release Table

Description

When used with a Data Table or Alternate Index that is open for write, the RL command changes the status of the table or index to open for read and releases the LOCK-LATCH if one has been set.

COBOL syntax

```
MOVE 'RL'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "RL", 2 );
TBLBASE( &tbParm, &tbCommArea );
```

Parameters

None

Explanation

RL can be used on any Data Table or Alternate Index with an open for write status.

A table that is released is then available for another application to open for write.

The RL command will not store the table. To store the table, an ST command must be issued before the RL command.

When the Data Table is released, any open Alternate Indexes are also released. An RL on an Alternate Index demotes the Alternate Index to open for read but leaves the Data Table open for write.

Return value

ERROR 0002 occurs when attempting to release a table that is not open.

ERROR 0073 occurs when attempting to release a table that is open for read.

Notes

The RL command should be used at the conclusion of the updating cycle if it is likely that users in other regions will want access to the table. As the table remains in memory it will not have to be loaded when other users begin using it and it is still accessible for read-only access in the region that has done the RL. Not having to reload the table is particularly beneficial with large tables.

Exceptions

None

See also

"9. LOCK-LATCH (8 bytes)" on page 146

Rename Table (RN)

Command title

Rename Table

Description

This command changes the name of all generations of a table in a tableBASE library.

COBOL syntax

```
MOVE 'RN'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA
                            NEW-TABLE-NAME
                            [PASSWORD].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "RN", 2 );
TBLBASE( &tbParm, &tbCommArea, pNewTableName,
         pWritePassword );
```

Parameters

NEW-TABLE-NAME

PASSWORD (optional)

Explanation

The library search list is used to find the first library containing the specified table.

Return value

ERROR 0003 subcode 1 can occur when attempting to rename a table that is open in a TSR.

ERROR 0030 occurs if an incorrect write PASSWORD is used.

ERROR 0072 occurs if the table is open for write in a different region, and the wait_for_enqueued_tables switch is set to N.

Notes

A table with the new name must not already exist in that library.

If a write password has been defined for the table, then a write password is require for this command.

Exceptions

None

See also

"Change Name (CN)" on page 70

Set Indirect (SI)

Command title Set Indirect

Description This command sets the criterion to be used when opening a table indirectly.

COBOL syntax

```
MOVE 'SI'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           INDIRECT-OPEN-CRITERION.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "SI", 2 );
TBLBASE( &tbParm, &tbCommArea, pInDirectOpenCriterion );
```

Parameters INDIRECT-OPEN-CRITERION

Explanation Set Indirect (SI) is used in conjunction with the Open Indirect option of the OR and OW commands. A common use of this feature is to choose a particular table from a set of different tables depending upon a date criterion supplied by the application.

To utilize Indirect Open, a table of table names (the primary table) must exist and be defined with only two fields: table-name (8 bytes) and lookup-value (up to 50 bytes) starting in location 9. The primary table must have a Sequential organization and the second field, lookup-value, must be defined as the key.

The SI command supplies an indirect open search criterion, which is saved for use with any subsequent indirect open command. The next occurring indirect open uses the table named in the open command as a primary table. The row in the primary table with the highest key less than or equal to the saved criterion is selected; its table-name field becomes the secondary table, which is the actual table opened by the indirect open command. The name of the secondary table is returned in the TABLE field of the xxxx-COMMAND-AREA.

To specify an indirect open, set the value I in the INDIRECT-OPEN field of the xxxx-COMMAND-AREA, before performing the open. An indirect open resets this field to blank.

The criterion is saved until reset by another SI command. It may be used with more than one indirect open.

Return value	In addition to possible open errors (see "Open for Write (OW)" on page 123, "Open for Read (OR)" on page 121), ERROR 0005 may occur if the search criterion are not matched.
Notes	None
Exceptions	None
See also	"Access a table indirectly" on page 220

Search by Key (SK)

Command title

Search by Key

Description

This command locates a row that has the same key as the one specified in the xxxx-KEY-AREA parameter.

COBOL syntax

```
MOVE 'SK'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           xxxx-KEY-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "SK", 2 );
TBLBASE( &tbParm, &tbCommArea, pKeyArea );
```

Parameters

xxxx-KEY-AREA

Explanation

If the row is found, COUNT is set to the subscript of the found row. The SK command does not retrieve a row from the table; a subsequent FETCH BY COUNT command will retrieve the new COUNT value set by SK.

Return value

If the row is found, FOUND is set to Y and COUNT is set to the subscript of the found row. If the requested row is not found in the table, FOUND is set to N and the COUNT will contain the appropriate subscript for a potential insertion.

Notes

Ideal for validation applications

Exceptions

None

See also

"3. FOUND (1 byte)" on page 144

"8. COUNT (fullword binary)" on page 146

Store Table (ST)

Command title

Store Table

Description

The ST command stores the table — table data and table definition — to a tableBASE library.

COBOL syntax

```
MOVE 'ST'                   TO xxxx-COMMAND.
CALL 'TBLBASE' USING        TB-PARM
                            xxxx-COMMAND-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "ST", 2 );
TBLBASE( &tbParm, &tbCommArea );
```

Parameters

None

Explanation

The ST command does not close the table. If an ST command addresses an Alternate Index, a new generation of the Data Table is actually stored instead; there is no effect on the Alternate Index itself.

Return value

ERROR 0018 occurs if there is insufficient space in the library.

ERROR 0019 occurs if the table is newly defined using DT or CN and a table with the same name is found in the library during the store.

Notes

tableBASE stores a new generation of a table before it deletes the oldest generation, if appropriate. Even if only one generation is to be kept, the new version of the table will be stored before the old space is released.

When stored, newly defined tables are automatically stored on the first library in the LIB-LIST (as set by the ML command) at the time of definition.

tableBASE does not permit a store of a newly defined table if the library contains a table with the same name. This prevents the inadvertent replacement of a previously defined table.

A table that has been read from a library will be stored into the library from which it was obtained, unless intentionally diverted by a DV or DW command.

Exceptions

None

See also

"Divert (Non-existing) Table (DV)" on page 87

"Divert (Existing) Table (DW)" on page 88

Un-allocate (UL)

Command title

Un-allocate

Description

This command will de-allocate a tableBASE library or any other file that was dynamically allocated to the region by an AL command.

COBOL syntax

```
MOVE 'UL'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           DDNAME.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "UL", 2 );
TBLBASE( &tbParm, &tbCommArea, pDDName );
```

Parameters

DDNAME—the library name that the application will use to refer to the allocated library

Explanation

None

Return value

The commands AL and UL will return an error code if they are used with DDNAMEs that are allocated through JCL or the TSO ALLOCATE command.

ERROR 0060, 0061, 0075 subcodes 8-10

See "tableBASE error codes" on page 408 for tableBASE error codes.

Notes

None

Exceptions

The UL command is valid only for single-task batch and IMS TM interfaces, and may be used only for files allocated by the AL commands.

See also

"Allocate (AL)" on page 63

Virtual Subsystem (VS)

Command title Virtual Subsystem

Description This command specifies the VTS-TSR as the target for all subsequent TBASEV and TBCALLV calls to tableBASE in the region.

COBOL syntax

```
MOVE 'VS'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           VTS-NAME.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "VS", 2 );
memcpy( tbParm.tbSubSystem, pSubSystemName, 4 );
TBLBASE( &tbParm, &tbCommArea );
```

Parameters VTS-NAME—the name of the VTS-TSR to use

Explanation None

Return value None

Notes It is recommended that you use TB-PARM to indicate the target VTS-TSR. This will override the VS.

To revert to the local TSR, issue a VS command with VTS-NAME set to blanks or low values.

This command overrides the VTSNAME parameter in TBOPT. The value to which VTS-NAME was set can be retrieved with the LV command.

Exceptions None

See also None

Eliminate Table (XT)

Command title Eliminate Table

Description This command deletes all generations of a table from the directory of the first of the libraries in the current search list in which the table is found.

COBOL syntax

```
MOVE 'XT'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA
                           [PASSWORD].
```

C syntax

```
memcpy( tbCommArea.tbCommand, "XT", 2 );
TBLBASE( &tbParm, &tbCommArea, pWritePassword );
```

Parameters PASSWORD (optional)

Explanation None

Return value ERROR 0009 subcode 1 occurs when the table is not found.

ERROR 0030 occurs when the wrong password is supplied.

Notes If a write password has been set, then a write password is required for this command.

Exceptions None

See also None

tableBASE Termination (XX)

Command title tableBASE Termination

Description This command is intended to be used at the end of tableBASE sessions.

COBOL syntax

```
MOVE 'XX'                  TO xxxx-COMMAND.
CALL 'TBLBASE' USING       TB-PARM
                           xxxx-COMMAND-AREA.
```

C syntax

```
memcpy( tbCommArea.tbCommand, "XX", 2 );
TBLBASE( &tbParm, &tbCommArea );
```

Parameters None

Explanation When this command is invoked the following activities are performed:

Any open tableBASE libraries are closed.

The TableSpace Report showing the statistics accumulated since the last strobe, or since the job was initiated if a full strobe interval has not elapsed, is closed.

Return value None

Notes None

Exceptions The XX command is not supported in CICS or DB2 SPAS.

In a multi-tasking batch environment, XX may only be issued by the persistent task, i.e., the task which first calls tableBASE, and which stays active until all tableBASE calls are completed.

See also None

tableBASE error codes

tableBASE messages and error codes are listed in Appendix E on page 407.

4
tableBASE parameter description

This chapter describes the parameter structures that are passed to tableBASE when calling the TBLBASE Application Programming Interface (API). See the individual command descriptions in Chapter 3 to determine parameter requirements for a given command. The complete list of tableBASE parameters includes:

ALT-DEFINITION
COMMAND-AREA
DATA-TABLE-NAME
DDNAME
DEFINITION-BLOCK
DIR-SPEC
GENERATION
INDIRECT-OPEN-CRITERION
KEY-AREA
LIB-LIST
LIB-SPACE
LIBRARY-ALLOC
LIST-BLOCK
NAME-AREA
NEW-GEN-NO
NEW-TABLE-NAME
PASSWORD
RELEASE LEVEL
ROW-AREA
STATUS-SWITCHES
TABLE-AREA
TABLE-STATS
TBACC-DEF
TB-PARM
VTS-DATA
VTS-NAME

Sample COBOL code is provided for each parameter, as it appears in your.prefix.TBASE.EDUCDATA(TBPARMS), the installation education dataset. Following recommended naming conventions for tableBASE application programs, any parameters which are dedicated to a single table should be prefixed by the table name in the program code. Accordingly, sample COBOL code for these parameters is prefixed here by xxxx to represent an application-specific table name.

At the end of this chapter, sample C code is provided for each parameter. Further information is available in "Programming in C with tableBASE" on page 177.

The following are the tableBASE parameters.

ALT-DEFINITION (12 bytes)

```
01 xxxx-ALT-DEFINITION.
   05 xxxx-ORG               PIC X.
   05 xxxx-METHOD            PIC X.
   05 xxxx-KEY-COUNT         PIC S9(4) COMP VALUE +1.
   05 xxxx-KEY-LOCATION      PIC S9(9) COMP.
   05 xxxx-KEY-SIZE          PIC S9(9) COMP.
```

This parameter contains all the values necessary for tableBASE to build an Alternate Index Definition. Defaults are invoked by setting the field to LOW-VALUES (hex zeroes) for any field or SPACES (blanks) for alpha fields. The ALT-DEFINITION parameter is used in commands CA and IA.

1. ORG (1 byte)

(ORGANIZATION.) Available table organizations are:

R or blank	=	Random (**default**)
U	=	User Sequence
S	=	Sequential
D	=	Descending Sequential
H	=	Hash

2. METHOD (1 byte)

(SEARCH-METHOD.) Search methods available are:

S	=	Serial (**default if Org. = R or U**)
Q	=	Queued Sequential
B	=	Binary (**default if Org. = S or D**)
C	=	Address Tree Binary
H	=	Hash (**default if Org. = H**)

3. KEY-COUNT (halfword binary)

For future use. Must be set to 1. The default = 1.

4. KEY-LOCATION (fullword binary)

The position of the key within the row. The default = 1.

5. KEY-SIZE (fullword binary)

Length of the key. The KEY SIZE must be from l to 256. The default = 1.

COMMAND-AREA (72 bytes)

This parameter is used by all commands.

```
 01  XXXX-COMMAND-AREA.
       05  XXXX-COMMAND                PIC XX   VALUE SPACES.
       05  XXXX-TABLE                  PIC X(8) VALUE 'XXXX'.
       05  XXXX-FOUND                  PIC X    VALUE SPACES.
       05  XXXX-INDIRECT-OPEN          PIC X    VALUE LOW-VALUES.
       05  FILLER                      PIC X    VALUE LOW-VALUES.
       05  XXXX-ABEND-OVERRIDE         PIC X    VALUE SPACES.
       05  XXXX-ERROR                  PIC S9(4) COMP VALUE +0.
       05  XXXX-COUNT                  PIC S9(9) COMP VALUE +0.
*   The LOCK must be specified for Online environments and VTS.
       05  XXXX-LOCK-LATCH             PIC X(8) VALUE SPACES.
*   Release 5.x command area extension
       05  XXXX-ROW-OVERRIDE-LENGTH    PIC S9(9) COMP VALUE +0.
       05  XXXX-ROW-ACTUAL-LENGTH      PIC S9(9) COMP VALUE +0.
       05  XXXX-FG-KEY-LENGTH          PIC S9(4) COMP VALUE +0.
       05  XXXX-FUNCTION-ID            PIC S9(4) COMP VALUE +0.
       05  XXXX-FUNCTION-AREA          PIC X(8)  VALUE LOW-VALUES.
       05  XXXX-DATE-AREA     REDEFINES XXXX-FUNCTION-AREA.
           10  XXXX-DATE               PIC X(8).
       05  FILLER                      PIC X(20).
       05  XXXX-RETURNED-ABS-GEN-NO    PIC S9(4) COMP VALUE +0.
       05  XXXX-ERROR-SUBCODE          PIC S9(4) COMP VALUE +0.
```


Note: Table access can be optimized by using a separate command area for each table accessed by an application. Instead of having to search a TSR directory each time to determine where the table resides in memory, tableBASE uses a reserved field in the command area (table handle) pointing to the position of the table in the TSR. By keeping a separate command area for each table accessed, this table handle is used to quickly locate a previously opened table.

1. COMMAND (2 bytes)

This is a 2-character code for the command to be performed by tableBASE.

2. TABLE (8 bytes)

This is the name of the target table for command processing. For the NX command it is used as a return field. For indirect Table Opens, it is used as both a send and return field.

3. FOUND (1 byte)

The FOUND field indicates that the specified row has been found. It provides relevant feedback for commands such as FK that must find a particular row in order to successfully execute. However, the value of this field is not relevant for commands that do not require a particular row found, such as IC.

All commands return a FOUND value of Y or N whether or not FOUND is applicable to the command. The default value is N.

The value of the FOUND field is relevant to retrieval commands (SK, FK, FC, FG, FN GF, GL, GN, and GP), update commands using keys (DK, RK, and IK), DC, RC, GD and NX. The FOUND field should be checked after calling tableBASE with any of these commands.

This field is used to indicate that the specified table row has been found with the exceptions listed in Table 4-1.

Table 4-1: Found byte

GD	FOUND is set to Y if the table is found.
NX	FOUND is set to Y if a table in the directory exists with a name greater than the input TABLE value.
DC	FOUND is set to Y if the row to be deleted was found.
RC	FOUND is set to Y if the row to be replaced was found.

4. INDIRECT-OPEN (1 byte)

A code of I is used on an Open command to indicate that a secondary table must be opened indirectly by searching a primary table of table names. Otherwise, this parameter should be initialized to blank or LOW-VALUES before the first use of the COMMAND-AREA, and should not be changed. The TABLE field of the COMMAND-AREA must be pre-set to identify the primary table for the indirect open. Once an Open command with the indirect option has been executed, the value I is reset to LOW-VALUES. After a successful Indirect Open operation, the selected secondary table name is placed in the TABLE field of the COMMAND-AREA.

For more information on Open Indirect, see "Set Indirect (SI)" on page 131.

5. FILLER (1 byte)

(RESERVED) This field is used internally by tableBASE. It should be initialized by the application to low values and must not be changed during program execution.

6. ABEND-OVERRIDE (1 byte)

This field is used on any command to temporarily override the current setting of the ABEND switch. If it is set to Y and an error occurs, the program (or transaction) will abend. If it is set to N and an error occurs, the program will not abend but the appropriate error code is returned in the ERROR field. If left blank, the action that will be taken is determined by the current setting of the ABEND Status switch (see "Change Status (CS)" on page 73). After the execution of any command, this field is reset to blank.

7. ERROR (halfword binary)

The ERROR code is set by every tableBASE command. Values other than zero denote a corresponding error condition.

Depending on an installation's default settings, tableBASE may abend on any error. However, when a CS (Change Status) command has been issued to disable Abend processing, tableBASE will continue processing when errors less than code 0100 or between 1000 and 1099 occur. This allows programs to continue regardless of table errors, but they must check the ERROR code after the call.

Version 6 introduced error subcodes (see "17. ERROR-SUBCODE (halfword binary)" on page 148).

8. COUNT (fullword binary)

This field refers to the subscript of the referenced row in a table. COUNT must be specified for FC, DC, RC, IC, GN, and GP commands. It is recommended that this value be set to zero when searching for the first occurrence of a partial key with the FG command. Thereafter, the FG command uses the count to position itself to the next row.

Valid values for COUNT:

- Generally, for all tables other than Hash tables, the COUNT is set to a value greater than zero and less than or equal to the actual number of rows in the table. There are exceptions: when an insert is performed the COUNT may exceed the actual number of rows by one; for the first FG and GN commands in a loop the COUNT may be zero; for the first GP command the COUNT may exceed the number of rows by one.
- For Hash tables, COUNT is set to a value greater than zero and less than or equal to the MAXIMUM table size. For DC and RC commands, COUNT may not point to an empty row.

If COUNT is outside of the permitted range:

- For an FC command the FOUND code will be set to N.
- For DC, RC, and IC commands an ERROR code will be returned.

COUNT is set by commands SK, FK, FG, GF, GN, GP, GL, DK, IK, and RK. If the row is found, COUNT will be set to the subscript of the row in the table. If the row is not found, COUNT will be set to the location for a potential insert operation.

COUNT is reset to zero by an explicit Table Open; it is reset to zero by an implicit open except for commands FC, IC, DC, RC, and DU.

9. LOCK-LATCH (8 bytes)

While a password can be used to provide security to a table in a library, the LOCK-LATCH provides control for updating a table that is open for write in a multi-user environment (CICS, IMS, multitasking batch, and tableBASE VTS).

To control the users or transactions that are authorized to perform updates in a multi-user environment after a table is open for write, place a password in the LOCK-LATCH field when Opening for Write using either an OW, DT or CN command. Only the update commands that have the same password in the LOCK-LATCH field are authorized to perform the update command.

This field is not edited and may contain any value. Leaving this field blank indicates a null LOCK-LATCH. So, although LOW-VALUES are considered a valid LOCK-LATCH, this is not recommended. **If the table is opened for write with a null LOCK-LATCH any update command can modify the table** regardless of the value in the LOCK-LATCH field.

A table that is opened for write with either a null LOCK-LATCH or a password initialized LOCK-LATCH cannot have the LOCK-LATCH modified in any way until the table is closed (CL) or released (RL).

If the LOCK-LATCH is forgotten or lost, the master password must be used to access the table. For more information, please contact your tableBASE administrator.

If the Data Table is closed or open for read, an Alternate Index that is opened for write with a LOCK-LATCH password will force the Data Table to adopt this LOCK-LATCH password and be opened for write. Once this Alternate Index is closed, and there are no other Alternate Indexes open, the Data Table's status is set to open for read and the LOCK-LATCH is released.

10. ROW-OVERRIDE-LENGTH (fullword binary)

This field denotes the relevant portion of the application's row area to be used for retrieval and update requests. If this field is zero, then the actual row length as specified in the table definition is used.

11. ROW-ACTUAL-LENGTH (fullword binary)

This is the length of a row in the table as returned by tableBASE.

12. FG-KEY-LENGTH (halfword binary)

A non-zero value in this field denotes the relevant length of a generic key parameter in a Fetch Generic request. If this field is zero, then the key parameter is assumed to contain a delimiter (usually an asterisk) that marks the end of the generic key.

13. FUNCTION-ID (halfword binary)

This field is used to distinguish between normal tableBASE processing and other specialized types of processing. Date-Sensitive Processing is the only special processing available at this time.

FUNCTION-ID = 0 for normal processing

FUNCTION-ID = 1 for Date-Sensitive Processing

FUNCTION-ID = 16 is used with the LT command. See "List Open Tables (LT)" on page 112.

14. FUNCTION-AREA: DATE (8 bytes)

The FUNCTION-AREA is used with the FUNCTION-ID. In this release, it is only implemented with the DATE field. The remainder of the FUNCTION-AREA is RESERVED.

This field is used for Date-Sensitive Processing if FUNCTION-ID = 1.The date specified here, as well as the dates specified in the table (effective and expiration date) must be of the same format: YYYYMMDD. The DATE field specifies the date against which the effective and expiration date will be checked (the baseline date), and it is used in conjunction with the row key to form the search key. If this field is spaces or low values, then the system's current date will be inserted here by tableBASE.

16. RETURNED-ABS-GEN-NO (halfword binary)

The absolute generation number of the table being accessed is returned to the user's extended command area and can be used to verify that the same generation of the table is being used from one call to the next.

The absolute generation number is returned whenever tableBASE processes an open table. This includes:

- All fetch commands
- All update commands
- Open commands
- Get Definition of an open table
- Commands when the table should be closed but is open, incorrectly or unexpectedly, e.g., Delete Generation, Change Generation, ReName, Define Table commands.

The absolute generation number is not returned by successful Close commands nor by successful 'closed table' commands, e.g. CG, DG, RN. XT.

17. ERROR-SUBCODE (halfword binary)

The ERROR-SUBCODE provides better diagnostics to pinpoint a reason for failure. If a task has the Abend Switch set on, a message with the error code and any related subcode is written to the job's JES log. The TBDRIVER (DK1TDRV) utility and the TBDR CICS transaction both display the error subcode. The ERROR-SUBCODE is dependent on the ERROR-CODE.

DATA-TABLE-NAME (8 bytes)

```
77     xxxx-DATA-TABLE-NAME                    PIC X(8).
```

This is the name of the Data Table for which an Alternate Index is being created, or for which an Alternate Index is being opened or invoked. The DATA-TABLE-NAME parameter is used by commands CA and IA. OR and OW have the TABLE field of the COMMAND-AREA set to the name of the Alternate Index name when opening Alternate Indexes.

DDNAME (8 bytes)

This is the DDNAME of a tableBASE library.

DEFINITION-BLOCK (256 bytes)

The DEFINITION-BLOCK is used to Define Tables (DT), Change Definitions (CD), Get Definition information (GD) and Dump Definition (DD). You may use the definition block in four ways:

- to define a new table using the DT command
- to modify an existing table using the CD command
- to obtain information about an existing table using the GD command
- to obtain minimal information about an existing table using the DD command.

Because of the multi-purpose uses of the DEFINITION-BLOCK, some fields are input, while others are output. Default values are invoked by setting a field to LOW-VALUES (for any field) or SPACES (for alpha fields).

```
01  XXXX-DEFINITION-BLOCK.
     05  XXXX-ORG                    PIC X     VALUE 'S'.
     05  XXXX-METHOD                 PIC X     VALUE 'B'.
     05  XXXX-INDEX                  PIC X     VALUE 'P'.
     05  XXXX-SMC                    PIC X     VALUE 'R'.
     05  XXXX-RPSWD                  PIC X(8)  VALUE SPACES.
     05  XXXX-WPSWD                  PIC X(8)  VALUE SPACES.
     05  XXXX-RSZ                    PIC S9(9) COMP VALUE +120.
     05  XXXX-KSZ                    PIC S9(9) COMP VALUE +11.
     05  XXXX-KLOC                   PIC S9(9) COMP VALUE +1.
     05  XXXX-ROWS                   PIC S9(9) COMP VALUE +500.
     05  XXXX-GENERATIONS            PIC S9(4) COMP VALUE +1.
     05  XXXX-EXP-FACT               PIC S9(4) COMP VALUE +200.
     05  XXXX-LO-DEN                 PIC S9(4) COMP VALUE +200.
     05  XXXX-HI-DEN                 PIC S9(4) COMP VALUE +500.
     05  FILLER                      PIC X(6)  VALUE LOW-VALUES.
     05  XXXX-DATE-TIME              PIC X(12) VALUE SPACES.
     05  XXXX-ABS-GEN-NO             PIC S9(4) COMP VALUE +0.
* following not returned by DD command
```



```
           05  XXXX-DATASET-NAME            PIC X(44).
           05  XXXX-REL-GEN-NO              PIC S9(4) COMP.
           05  XXXX-GENS-PRESENT            PIC S9(4) COMP.
           05  XXXX-ROWS-AT-EXPAND          PIC S9(8) COMP.
           05  XXXX-DDNAME                  PIC X(8).
           05  XXXX-DATA-TABLE              PIC X(8).
           05  XXXX-OPEN-STATUS             PIC X.
           05  XXXX-ALTS-INVOKED            PIC X.
       *   Release 5.x additions follow
           05  XXXX-VIEW-VERSION            PIC X.
           05  FILLER                       PIC X.
           05  XXXX-USERID                  PIC X(8).
           05  XXXX-VIEW-NAME               PIC X(8).
           05  XXXX-VIEW-DATE               PIC X(12).
           05  XXXX-USER-COMMENTS           PIC X(16).
       *   Release 6.0.3 / 6.1.0 addition follows
           05  XXXX-VTSNAME                 PIC X(08).
           05  FILLER                       PIC X(68).
```

Note: The fields VIEW-VERSION through USER-COMMENTS were added in Version 5. These fields are not captured or displayed if the TB-FORMAT field in the TB-PARM parameter is A, nor if the TB-PARM field is not supplied.

1. ORG (1 byte)

(ORGANIZATION.) Available table organizations are:

- R or blank—Random (**default**)
- U—User Sequence
- S—Sequential
- D—Descending Sequential
- H—Hash

2. METHOD (1 byte)

(SEARCH-METHOD.) Search methods available are:

- S—Serial (**default if Org. = R or U**)
- Q—Queued Sequential
- B—Binary (**default if Org. = S or D**)
- C—Address-Tree Binary
- H—Hash (**default if Org. = H**)

3. INDEX (1 byte)

This field is no longer meaningful and is kept for backwards compatibility. Previous to Version 6 tables could be set up without a primary Index (a True table). Valid values are:

- P—Pointer table
- T—True table

With Version 6, all tables are Pointer tables. The term pointer table indicates the combination of the Index and a Data Table. The use of an Index improves efficiency of insert and delete operations in sequential tables and reduces memory requirements for Hash tables.

4. SMC (1 byte)

(STORAGE-MODE-CODE)—controls how a table is to reside in memory. The default is for tables to be fully resident in memory. The values X and A are returned by a GD command if the table is an Alternate Index. They may not be set in DT or CD commands. The SMC may have the following settings:

- R—Resident Table Default
- X—Alternate Definition (**internal use only**)
- A—Invoked Alternate (**internal use only**).

5. RPSWD (8 bytes)

(READ-PASSWORD) A read password protects a table from being opened for either read or write from a tableBASE library. When a Get Definition (GD) command is issued, the presence of a READ-PASSWORD will be signalled by a value of "XXXXXXXX".

To remove a password, open the table for write, use the Change Table Definition (CD) command, and ensure that the first byte of the READ-PASSWORD contains an asterisk (*). This causes the READ-PASSWORD to be set to spaces which indicates a null password.

The default for the READ-PASSWORD is a null password (spaces).

6. WPSWD (8 bytes)

(WRITE-PASSWORD.) A write password protects a table from being opened for write from a tableBASE library. This password will be checked in the processing of an Open for Write (OW), Delete Generation (DG), Change Generations (CG), Eliminate Table (XT), Rename Table (RN) or Divert (Existing) Table (DW) command. When a Get Table Definition (GD) command is issued, the presence of a WRITE-PASSWORD will be signalled by a value of "XXXXXXXX".

To remove a password open the table for write and use the CD command and ensure that the first byte of the WRITE-PASSWORD contains an asterisk (*). This causes the WRITE-PASSWORD to be set to spaces which indicates a null password.

The default for the WRITE-PASSWORD is the READ-PASSWORD. This means that if you are attempting to open for write a table that has a read password but no write password, you must supply the READ-PASSWORD.

7. RSZ (fullword binary)

(ROW-SIZE) This specifies the length of each row in the table. The ROW-SIZE must be from 1 to 32,767.

The default ROW-SIZE is 1.

For an Alternate Index, ROW SIZE will be shown as zero until the table is opened.

8. KSZ (fullword binary)

(KEY-SIZE) This is the length of the key. The KEY-SIZE must be from 1 to 256.

The default is ROW-SIZE or 256, whichever is smaller.

9. KLOC (fullword binary)

(KEY-LOCATION) The position of the key in the ROW-AREA. The entire key must be within the ROW-AREA. The default is 1.

Note: Key location in tableBASE is relative to 1, not zero.

10. ROWS (fullword binary)

(NUMBER-OF-ROWS) An estimate of the number of rows that will initially be loaded into the table:

- default NUMBER-OF-ROWS for tables depends on the row-size and table organization
- size of a table is limited only by the amount of memory available.

Note: The space required for a data table =
Row-Space + Index-Space + Definition-Space
The space required for an alternate index =
Index-Space + Definition-Space
Row-Space = NUMBER-OF-ROWS * row size rounded up to a multiple of 4K
Index-Space for a non-hash index =
NUMBER-OF-ROWS * 8 * (1 + expansion factor/1000)


```
                rounded up to a multiple of 4K
Index-Space for a hash index  =
      NUMBER-OF-ROWS * 8 * lower density/1000
                rounded up to a multiple of 4K
Definition-Space  =  4K
```

After a GD command, this field contains the actual number of rows in the table.

11. GENERATIONS (halfword binary)

This is the maximum number of table generations to be kept on the library. Its value must be from l to 9.

The default is 1.

12. EXP-FACT (halfword binary)

(EXPANSION-FACTOR) The Expansion Factor controls the amount of additional memory that will be obtained to expand non-hash tables. The Expansion Factor is supplied as an integer from 1 to 999 representing the number of tenths of a percent to be expanded. An Expansion Factor of 250 will be treated as 25%.

The default EXPANSION-FACTOR is 200 (20%).

For tables with a hash organization the Expansion Factor is used for the data portion; the Low Density and High Density are used for the Index portion.

13. LO-DEN (halfword binary).

(LOW-DENSITY) This is the density to which a hash table will be set when reorganized. Reorganization is done automatically when the HIGH-DENSITY is exceeded. LOW-DENSITY is an integer from 1 to an upper limit representing the density in thousandths.

The upper limit is the HASH_LOW_DEN_LIM option value established at installation or overriden in TBOPT dataset at execution time. LOW-DENSITY must be no greater than 2/3 of HIGH-DENSITY. LOW-DENSITY will be adjusted if it exceeds these limits.

14. HI-DEN (halfword binary).

(HIGH-DENSITY) This is the value for the highest density a hash table may have. When an insertion of a row into a hash table would cause the HIGH-DENSITY to be exceeded, the table will be reorganized to match the LOW-DENSITY specification. HIGH-DENSITY is an integer from 1 to an upper limit representing the density in thousandths.

The upper limit is the HASH_HI_DEN_LIM option value established at installation or overriden in TBOPT dataset at execution time. HIGH-DENSITY must be at least 1.5 times LOW-DENSITY. HIGH-DENSITY will be adjusted if it exceeds these limits.

15. FILLER (6 bytes)

16. DATE (12 bytes)

The date and time that a new table was defined, or when this generation of the table was stored. While the table is in storage, this value is not updated. The format is: YYYYMMDDHHMM (year, month, day, hour, minute), and is provided by a GD command. This field is zero for a new table until it is stored. For a temporary table, this field will reflect the time that it was created in the TSR.

17. ABS-GEN-NO (halfword binary).

(ABSOLUTE-GENERATION-NUMBER) The absolute generation number of the table for which a GD command was issued. This field will be zero if the table has never been stored. For more information see "Generations" on page 49.

Note: In version 5, when an alternate index is defined in memory only, the GD command showed the number of generations as 0; in Version 6, the number of generations is shown as 1.

18. DATASET-NAME (44 bytes)

The full Dataset Name of the tableBASE library on which the table resides. This field may contain spaces if the table was defined with a DT command and never stored.

19. REL-GEN-NO (halfword binary)

(RELATIVE-GENERATION-NUMBER) The relative generation number of the table will have a value ranging from zero to -8. If the table has never been stored the value will be zero. For more information see "Generations" on page 49.

20. GENS-PRESENT (halfword binary)

(GENERATIONS-PRESENT) The number of generations of this table that exist at the time of the GD command. This will be a value from 0 to 9. Zero indicates that the table has never been stored.

21. ROWS-AT-EXPAND (fullword binary)

(MAXIMUM-ROWS-FOR-SPACE-ALLOCATED) This is the number of rows that the table can hold, when open, before either the row space or the index must be expanded.

22. DDNAME (8 bytes)

This is the DDNAME of the tableBASE library. See "18. DATASET-NAME (44 bytes)" .

23. DATA-TABLE (8 bytes)

(DATA-TABLE-NAME) This is the name of the Data Table (also known as the base or primary table) when the Definition Block is for an Alternate Index.

24. OPEN-STATUS (1 byte)

This field will contain one of three values:

- R—Table open for Read in the TSR
- W—Table open for Write in the TSR
- X—The definition was obtained from the tableBASE library. This value is obtained if the table is not open or if the GENERATION parameter was provided with the GD command.

25. ALTS-INVOKED (1 byte)

(ALTERNATES-INVOKED) If there are invoked Alternate Indexes for this table, this field will be Y, otherwise it will be N.

26. VIEW-VERSION (1 byte)

This field is used only by the optional tableBASE component tablesONLINE/CICS and identifies the release level of the View table (when it is a View) used to display a Data Table in an online environment. This field is automatically updated by tablesONLINE when it converts the View to reflect Version 5 and higher features.

27. FILLER (1 byte)

(RESERVED) Reserved for internal use.

28. USERID (8 bytes)

This field contains the UserID of the user who last updated or stored the table.

29. VIEW-NAME (8 bytes)

This field is used only by the optional tableBASE component tablesONLINE/CICS and is the name of the View used to display the table in an online environment.

30. VIEW-DATE (12 bytes)

This field is used only by the optional tableBASE component tablesONLINE/CICS and is the creation date of the View used to display the table in an online environment.

31. USER-COMMENTS (16 bytes)

This field contains user comments.

32. VTSNAME (8 bytes)

This field contains the name of the VTS-TSR in which the table resides (if applicable).

33. FILLER (68 bytes)

(RESERVED) Reserved for future expansion.

DIR-SPEC (9 bytes)

This parameter specifies which tables are to be included in the library directory list for an LD command.

```
01 DIR-SPEC.
   05 TABLE-NAME-MASK     PIC X(8).
   05 DIRTYPE             PIC X.
```

1. TABLE-NAME-MASK (8 bytes)

The mask identifies a table name, or a partial table name terminated by a wild card character (*), to be included in the directory list.

2. DIRTYPE (1 byte)

This field specifies a category of tables to be included in the directory list:

- **V**—View tables (based on tablesONLINE definition); all table names beginning with x'80' to x'BF', including lower-case characters
- **D**—Latest generation of the Data Tables (based on tablesONLINE definition); all table names beginning with x'C0' to x'FF', including lower-case characters, including all upper-case characters and numbers
- **T**—All generations of all tables—this is the default value.

GENERATION (fullword binary)

The table generation number to be used with the command. If this number is positive, it indicates an absolute generation number; if it is zero or negative, it indicates a relative generation (relative to the most recent generation stored). A valid generation number is between 1 and 255 or 0 and -8.

```
77  xxxx-GENERATION          PIC S9(9) COMP.
```

INDIRECT-OPEN-CRITERION (50 bytes)

The selection criterion to be used for indirect table access. The indirect parameter is used to set the criterion for a subsequent indirect open request. At open time, tableBASE searches the primary table for the row with the largest key equal to or less than the value of the INDIRECT-OPEN-CRITERION parameter. The table name in that row is the indirect (secondary) table that will be accessed.

```
77  xxxx-INDIRECT-OPEN-CRITERION             PIC X(50).
```


KEY-AREA (Table Specific)

An area suitably sized to contain the key to be used in a Fetch, Delete, Insert, or Replace. It may be within the ROW-AREA described below or it may be defined as a separate area. Maximum length is 256.

```
77  xxxx-KEY-AREA                          PIC X(nnn).
```

LIB-LIST (80 bytes)

The DDNAMEs of up to 10 tableBASE libraries in the order in which they are to be searched.

```
01 LIB-LIST.
   05 LIB-1              PIC X(8) VALUE 'TESTLIB'.
   05 LIB-2              PIC X(8) VALUE 'MAINLIB'.
   05 LIB-3              PIC X(8) VALUE 'VTS:DKL1'.
   05 LIB-4              PIC X(8) VALUE SPACES.
   05 LIB-5              PIC X(8) VALUE SPACES.
   05 LIB-6              PIC X(8) VALUE SPACES.
   05 LIB-7              PIC X(8) VALUE SPACES.
   05 LIB-8              PIC X(8) VALUE SPACES.
   05 LIB-9              PIC X(8) VALUE SPACES.
   05 LIB-10             PIC X(8) VALUE SPACES.
```

The unused low order portion of LIB-LIST must be set to low values or spaces. A VTS-TSR may also be included in the search list by specifying the name of the VTS-TSR, prefixed by the VTSPREFIX in TBOPT, instead of a library DDNAME. For example, if VTSPREFIX is not specified in TBOPT and defaults to VTS:, and the VTS name is DKL1, then a LIB-LIST value of VTS:DKL1 could be used; or if VTSPREFIX is set to V: in TBOPT, and the VTS name is VTS1234, then a LIB-LIST value of V:VTS1234 could be used.

tableBASE stops processing the list when it reaches an entry with all low-values or all spaces.

LIB-SPACE (8 bytes)

Two fullword binary counters which will contain the number of library blocks defined and the number of blocks remaining. These values are returned for the DL and NX commands.

```
01  LIB-SPACE.
    05 LIB-BLOCKS-ALLOCATED        PIC S9(9) COMP.
    05 LIB-BLOCKS-REMAINING        PIC S9(9) COMP.
```

where:

LIB-BLOCKS-ALLOCATED (fullword binary) is the number of blocks defined for the library.

LIB-BLOCKS-REMAINING (fullword binary) is the number of unused blocks on the library.

LIBRARY-ALLOC (45 bytes)

The information for dynamically allocating a tableBASE library or another dataset.

```
01  LIBRARY-ALLOC.
    05 LIBRARY-SHARE-STATUS            PIC X VALUE 'S'.
    05 LIBRARY-DATASET-NAME            PIC X(44)
                              VALUE 'YOUR.PREFIX.TBASE.MAINLIB'.
```

where:

LIBRARY-SHARE-STATUS (l byte) indicates how the allocated library is to be shared, and is equivalent to the DISP parameter in MVS JCL.

- O—for exclusive access (OLD)
- S—for shared access (SHR).

LIBRARY-DATASET-NAME (44 bytes) is the name of the dataset to be allocated.

LIST-BLOCK (88 bytes)

This parameter is used by the LT command and contains information about the current and past use of the TSR. The first three fields are input fields and control the information to be placed in the optional TABLE-STATS parameter for the LT command. The rest of the fields are output fields and describe usage statistics about the TSR.

The amount of information returned in the LIST-BLOCK is controlled by the FUNCTION-ID in the COMMAND-AREA. If the FUNCTION-ID is 0, then only the first eleven fields of the LIST-BLOCK will be used; i.e., after the first three input fields, only the fields up to and including LIST-STROBE will be returned. If the FUNCTION-ID is 16, then all fields (88 bytes) will be used.

```
 01 LIST-BLOCK.
* Values set for use by LT command
     05 LIST-FROM              PIC S9(9) COMP VALUE +1.
     05 LIST-REQD              PIC S9(9) COMP VALUE +96.
     05 LIST-SIZE              PIC S9(9) COMP VALUE +80.
* Standard values returned by LT command
     05 LIST-TOTAL             PIC S9(9)  COMP.
     05 LIST-RETURNED          PIC S9(9)  COMP.
     05 LIST-TSR-HW            PIC S9(9)  COMP.
     05 LIST-OPEN-HW           PIC S9(9)  COMP.
     05 LIST-OPEN-NOW          PIC S9(9)  COMP.
     05 LIST-TSR-NOW           PIC S9(9)  COMP.
     05 LIST-TSR-SZ            PIC S9(9)  COMP.
     05 LIST-STROBE            PIC S9(9)  COMP.
* Extra values returned when function-id is 16 (x'10')
     05 LIST-TOTAL-HWM         PIC S9(9)  COMP.
     05 LIST-TOTAL-CALLS       PIC S9(18) COMP.
     05 LIST-MAX-TBLS          PIC S9(9)  COMP.
```



```
05 FILLER                PIC X(16).
05 LIST-TSR-AVAIL        PIC S9(9)  COMP.
05 FILLER                PIC X(8).
```

1. LIST-FROM (fullword binary)

The starting point in the list of OPEN tables in the current TSR, relative to 1, for rows to be returned to the TABLE-STATS parameter.

2. LIST-REQD (fullword binary)

The number of rows to be returned to the TABLE-STATS parameter.

3. LIST-SIZE (fullword binary)

The size of each row to be returned to the TABLE-STATS parameter. Each row contains statistics about an open table. This field gives the programmer the opportunity of limiting the output of table statistics.

4. LIST-TOTAL (fullword binary)

The total number of tables open.

5. LIST-RETURNED (fullword binary)

The number of table statistics rows returned from this LT request to the TABLE-STATS parameter. tableBASE will put -1 (negative one) in this field if it finds a problem with LIST-FROM, LIST-REQD- or LIST-SIZE.

6. LIST-TSR-HW (fullword binary)

The largest amount ("high water mark") of the current TSR used since it was started.

7. LIST-OPEN-HW (fullword binary)

The largest aggregate size of all concurrently open tables ("high water mark") since the TSR was started.

8. LIST-OPEN-NOW (fullword binary)

The amount of space currently used by all open tables in the current TSR.

9. LIST-TSR-NOW (fullword binary)

The amount of space currently used by the open tables that are currently in the TSR.

10. LIST-TSR-SZ (fullword binary)

The size of the TSR, in bytes.

11. LIST-STROBE (fullword binary)

The Strobe Interval—the number of tableBASE accesses required to be executed before automatically generating a snapshot of the above information to the TSR.

12. LIST-TOTAL-HWM (fullword binary)

The High Water Mark for the number of open tables.

13. LIST-TOTAL-CALLS (doubleword binary)

The total call count for all tables in the TSR. This field is a 64-bit signed integer to accommodate very large call counts.

14. LIST-MAX-TBLS (fullword binary)

The maximum number of tables allowed in the TSR.

15. FILLER (16 bytes)

Reserved for future use.

16. LIST-TSR-AVAIL (fullword binary)

The number of 4096-byte blocks available in TSR.

17. FILLER (8 bytes)

Reserved for future use.

NAME-AREA (256 bytes)

This area is used by the BN command to return the client name and address.

```
77    NAME-AREA                 PIC X(256).
```

NEW-GEN-NO (fullword binary)

The new number of generations to be kept. Its value may be from 1 to 9. If less than the number of generations actually on the library for this table, an appropriate number of the oldest generations will be deleted.

```
77  NEW-GEN-NO                 PIC S9(9) COMP.
```

NEW-TABLE-NAME (8 bytes)

The new name for a table that is being renamed in a TSR (CN) or a library (RN).

```
77  NEW-TABLE-NAME             PIC X(8).
```

PASSWORD (8 bytes)

The password with which this table is to be opened. If the command is Open for Read (OR), either the read or write password may be supplied. Other commands requiring a password should supply the write password.

If the table has only a read password defined, then the write password defaults to the read password, and it must be supplied for the table to be opened for write. The password is given in the PASSWORD parameter.

```
77  PASSWORD                   PIC X(8).
```

Note: The LOCK-LATCH provides some protection for tables once they are opened (see page 146).

RELEASE-LEVEL (16 bytes)

This area is used by the BN command to return the release level.

```
77    RELEASE-LEVEL             PIC X(16).
```

ROW-AREA (Table Specific)

An area large enough to contain one full row of the table. This area will be used to receive the contents of the appropriate row from the table after a Fetch, Get, or Delete command. If there is a likelihood that the row size will increase with new fields, it is prudent to

define this area a little larger to accommodate this growth. Then when growth occurs, the programs that don't require the additional data do not require a recompile.

```
01 xxxx-ROW-AREA.
   05 xxxx-ROW-KEY                 PIC X(nn).
   05 xxxx-ROW-DATA                PIC X(nn).
```

The Insert and Replace commands will take the data from the ROW-AREA to insert or replace a row in the table.

The number of bytes transferred between the ROW-AREA and the table may be less than or equal to the actual length of a row in the table, depending on the value of the ROW-OVERRIDE-LENGTH field in the COMMAND-AREA.

Maximum row size is 32767 bytes; minimum row size is 1 byte.

STATUS-SWITCHES (8 bytes)

A series of one-byte status switches for listing (LS) or altering (CS) the operational characteristics of tableBASE. For additional information, see "List Status (LS)" on page 111 and "Change Status (CS)" on page 73.

```
01  STATUS-SWITCHES.
   05 ST-ABEND                      PIC X VALUE SPACES.
   05 ST-WAIT                       PIC X VALUE SPACES.
   05 ST-HASH-EMPTIES-RETURNED      PIC X VALUE SPACES.
   05 ST-DEFAULT-OPEN               PIC X VALUE SPACES.
   05 ST-TRACE                      PIC X VALUE SPACES.
   05 ST-RESERVED                   PIC X(3) VALUE SPACES.
```

1. ABEND (1 byte)

Y—If Abend processing is to be performed on tableBASE errors 0001 to 0099 and 1000 to 1099

N—If Abend processing is not to be performed on tableBASE errors 0001 to 0099 and 1000 to 1099

blank—If Abend handling is to remain as previously set

2. WAIT (1 byte)

Y—if tableBASE is to wait for tables which are enqueued

N—if tableBASE is not to wait for such enqueued tables. In this case tableBASE will Abend with a User 0072 if Abend processing is enabled, or will return an error code of 0072 if Abend processing has been disabled.

blank—if Wait handling is to remain as previously set

Note: This switch should normally be set to N in online and multitasking environments. Otherwise, a long running batch update or tablesONLINE edit session may cause an online transaction to wait for an excessive period of time.

3. HASH-EMPTIES-RETURNED (1 byte)

Y—if tableBASE is to return empty rows from hash tables

N—if tableBASE is to suppress the return of empty rows from hash tables for GF, GL, GN and GP commands.

blank—if Empty Rows handling is to remain as previously set

4. DEFAULT-OPEN (1 byte)

Y—if tableBASE is to automatically open tables for read

N—if tableBASE is to suppress automatic opens (an explicit OR or OW command must be issued to open a table)

blank—if Default Open handling is to remain as previously set

5. TRACE (1 byte)

Y—if tableBASE is to maintain a Trace of the last ten commands per thread

N—if tableBASE is not to maintain a Trace table

blank—if Trace handling is to remain as previously set

6. FILLER (3 bytes)

(RESERVED) Reserved for future use.

TABLE-AREA

An area sufficiently large to receive the contents of the table being dumped with a DU command. The companion parameter TBACC-DEF will receive the tableBASE Access Routine Definition on a DU command.

```
01 xxxx-TABLE-AREA.
   05  xxxx-TABLE-ROW           PIC X(nnn)  OCCURS mmm.
```


TABLE-STATS

An area sufficiently sized to receive the table statistics from the LT command; it will return up to 80 bytes of table statistics. The copybook OCCURS clause must be changed to match the number of rows specified by the LIST-REQD field of the LIST-BLOCK parameter. The size of each table related statistics row (specified by LIST-SIZE) determines how many fields within the row are returned.

```
01  TABLE-STATS.
    05 LIST-TABLE-ENTRY             OCCURS 36 TIMES.
       10 TABLE-NAME                PIC X(8).
       10 TABLE-OPEN-STATUS         PIC X.
       10 TABLE-LOCAL-VTS           PIC X.
       10 TABLE-ALT-INVOKED         PIC X.
       10 FILLER                    PIC X.
       10 TABLE-CALLS               PIC S9(9) COMP.
       10 TABLE-SIZE                PIC S9(9) COMP.
       10 TABLE-ROWS                PIC S9(9) COMP.
       10 TABLE-RWS-BF-EXP          PIC S9(9) COMP.
       10 TABLE-DATATBL-VTSNAME     PIC X(8).
       10 TABLE-UPDATE-CALLS-TRUNC  PIC S9(9) COMP.
       10 TABLE-DATE-TIME           PIC 9(12).
       10 FILLER                    PIC S9(4) COMP.
       10 TABLE-CALLS               PIC S9(18) COMP.
       10 TABLE-UPDATE-CALLS        PIC S9(18) COMP.
       10 TABLE-VTSNAME             PIC X(8).
```

1. TABLE-NAME (8 bytes)

The name of the open table for which the statistics are supplied.

2. TABLE-OPEN-STATUS (1 byte)

The open status of the table: W for write, R for read.

3. TABLE-LOCAL-VTS (1 byte)

This field only has meaning when LT is issued against a local TSR. It indicates whether the table physically resides in the local TSR (L), or is a logical link to a table residing in a VTS-TSR (V).

4. TABLE-ALT-INVOKED (1 byte)

This indicates whether there are Alternate Indexes invoked (Y or N) for this table.

5. FILLER (1 byte)

(RESERVED) Reserved for internal use.

6. TABLE-CALLS (fullword binary)

The total number of calls made against the table since it was last opened.

7. TABLE-SIZE (fullword binary)

The size of the table in bytes.

8. TABLE-ROWS (fullword binary)

The number of rows currently in the table.

9. TABLE-RWS-BF-EXP (fullword binary)

The lesser of: (a) the number of rows, or (b) the number of index entries, that the current table can contain before automatic expansion. Expansion occurs when this number is exceeded.

10. TABLE-DATATBL-VTSNAME (8 bytes)

The name of the Data Table if this entry describes an Alternate Index or the LIB-LIST entry name in the form VTS:xxxx if this entry describes a Linked table.

11. TABLE-UPDATE-CALLS-TRUNC (fullword binary)

Number of updates to this table.

12. TABLE-DATE-TIME (12 bytes)

The date and time that the table was last stored to a library.

13. FILLER (4 bytes)

(RESERVED) Reserved for internal use.

14. TABLE-TOTAL-CALLS (doubleword binary)

The total number of calls made to this table since it was last opened.

15. TABLE-UPDATE-CALLS (doubleword binary)

The total number of updates to the table since it was last opened.

16. TABLE-VTSNAME (8 bytes)

The name of the VTS-TSR if this entry describes a Linked table.

TBACC-DEF (29 bytes)

This parameter is used to download tableBASE table definition data for subsequent use by TBACC. When a program requires internal use of a table maintained on a tableBASE library, the DU command copies this table to a program area. The final 108 bytes are reserved for additional area used by TBACC commands. (For further information, see "Subroutine TBACC" on page 289).

```
01 xxxx-TBL-DEF.
   05  xxxx-ROWS                PIC S9(8) COMP VALUE +0.
   05  xxxx-SIZE                PIC S9(8) COMP VALUE +100.
   05  xxxx-KEYSIZE             PIC S9(8) COMP VALUE +7.
   05  xxxx-KEYLOC              PIC S9(8) COMP VALUE +4.
   05  xxxx-MAX                 PIC S9(8) COMP VALUE +1000.
   05  xxxx-SUB                 PIC S9(8) COMP VALUE +0.
   05  xxxx-ORG                 PIC X VALUE 'S'.
   05  xxxx-FOUND               PIC X.
   05  xxxx-OVFLOW              PIC X.
   05  xxxx-ACTION              PIC X VALUE 'S'.
   05  xxxx-METHOD              PIC X VALUE 'C'.
   05  RESERVED                 PIC XXX VALUE LOW-VALUES.
   05  RESERVED                 PIC X(108) VALUE LOW-VALUES.
```

TBINDEX-DEF (32 bytes)

This parameter is used to download tableBASE table definition data for subsequent use by TBINDX. When a program requires internal use of a table maintained on a tableBASE library, the DU command copies this table to a program area. The final 220 bytes are reserved for additional area used by TBINDX commands. (For further information, see "Subroutine TBINDX" on page 299).

```
01  XXXX-TBI-DEF.
    05  XXXX-ROWS                  PIC S9(8) COMP VALUE +0.
    05  XXXX-SIZE                  PIC S9(8) COMP VALUE +100.
    05  XXXX-KEYSIZE               PIC S9(8) COMP VALUE +7.
    05  XXXX-KEYLOC                PIC S9(8) COMP VALUE +4.
    05  XXXX-MAX                   PIC S9(8) COMP VALUE +1000.
    05  XXXX-SUB                   PIC S9(8) COMP VALUE +0.
    05  XXXX-ORG                   PIC X     VALUE 'S'.
    05  XXXX-FOUND                 PIC X.
    05  XXXX-OVFLOW                PIC X.
    05  XXXX-ACTION                PIC X     VALUE 'S'.
    05  XXXX-METHOD                PIC X     VALUE 'C'.
    05  FILLER                     PIC X(3)  VALUE LOW-VALUES.
    05  XXXX-PRIMARY-SUB           PIC S9(8) COMP.
    05  XXXX-WORK-AREA             PIC X(220) VALUE SPACES.
    05  FILLER REDEFINES XXXX-WORK-AREA.
        10  XXXX-TAG-MAX             PIC S9(8) COMP VALUE +1000.
        10  FILLER                   PIC X(216) VALUE SPACES.
```


TB-PARM (64 bytes)

This parameter is a general communication area for tableBASE. It contains tableBASE release information, Date-Sensitive Processing controls, and pointers to transaction specific save areas for online environments.

```
 01  TB-PARM.
      10  TB-PARM-ID                  PIC X(2)  VALUE 'TB'.
      10  FILLER                      PIC X(2)  VALUE LOW-VALUES.
      10  TB-VERSION                  PIC X     VALUE '5'.
      10  TB-FORMAT                   PIC X     VALUE '0'.
      10  FILLER                      PIC X(10) VALUE LOW-VALUES.
*  TPVM valid only with Version 6.1.1
      10  TB-TPVM                     PIC X(8)  VALUE LOW-VALUES.
      10  TB-SUBSYSTEM                PIC X(8)  VALUE LOW-VALUES.
      10  FILLER                      PIC X(4)  VALUE LOW-VALUES.
      10  TB-TURBO-ANCHOR             PIC X(8)  VALUE LOW-VALUES.
      10  FILLER                      PIC X(20) VALUE LOW-VALUES.
```

1. TB-PARM-ID (2 bytes)

The literal "TB" identifies this parameter as the TB-PARM communications area.

2. FILLER (2 bytes)

(TB-RESERVED) This field is reserved for internal use.

3. TB-VERSION (1 byte)

This field identifies the major tableBASE format level. If you wish to use the TBLBASE interface, this field must be set to 5. This remains unchanged for Version 6.

4. TB-FORMAT (1 byte)

This field identifies the format of the COMMAND-AREA as a 20-byte or a 28-byte Release 4.x format, or a 72-byte Release 5.0 and above format.

TB-FORMAT = 0 for a 72-byte extended COMMAND-AREA

TB-FORMAT = A for a 20-byte or a 28-byte Release 4.x COMMAND-AREA.

Note: A 20-byte command area is acceptable when TB-FORMAT = A as long as the application does not reference a VTS-TSR, in which the command area is either of a 72-byte (Release 5.x format) or a 28-byte (Release 4.x) format.

5. FILLER (10 bytes)

(TB-RESERVED) This field is reserved for internal use.

6. TB-TPVM (8 bytes)

Specifies which VTS Manager (TPVM) to use for VTS access, if applicable. This parameter is valid for Release 6.1.1 and above, with the tableBASE Process Manager product.

7. TB-SUBSYSTEM (8 bytes)

Entering a name of a VTS-TSR in this field completely bypasses the use of the local TSR in favor of the VTS-TSR. This should be set to low values or spaces to indicate a local TSR.

Note: With Release 6.0.3 and above, a VTS-TSR name can be up to 8 characters. Previous releases supported only 4-character VTS names. For compatibility with existing 4-character VTS names, if the last 4 characters of an 8-character VTS name are low-values (binary zeroes), they will be ignored when determining which VTS-TSR to access.

8. FILLER (4 bytes)

(TB-RESERVED) This field is reserved for internal use.

9. TB-TURBO-ANCHOR (8 bytes)

This field is used for high performance table access. It must be set to low-values at the beginning of each thread of execution (CICS transaction, IMS TM message, DB2 Stored Procedure, batch job, or task in a multiprocessing batch job) prior to the first call to tableBASE, and is thereafter reserved for internal use.

10. FILLER (20 bytes)

(TB-RESERVED) This field is reserved for internal use.

VTS-DATA (60 bytes)

This parameter is used to return VTS-TSR name information from tableBASE when an LV command is issued.

```
01  VTS-DATA.
     10  FILLER                      PIC X(8).
     10  VTS-VTSFIRST                PIC X(8).
     10  VTS-VTSLAST                 PIC X(8).
     10  VTS-VTSNAME                 PIC X(8).
     10  FILLER                      PIC X(16).
*  TPVM valid only with Version 6.1.1
     10  VTS-TPVM                    PIC X(8).
     10  VTS-PREFIX                  PIC X(4).
```

1. FILLER (8 bytes)

This field is reserved for internal use (Release 6.0.3).

2. VTS-VTSFIRST (8 bytes)

This field returns the VTSFIRST parameter from TBOPT. For information on this parameter, see Appendix C on page 389.

3. VTS-VTSLAST (8 bytes)

This field returns the VTSLAST parameter from TBOPT. For information on this parameter, see Appendix C on page 389.

4. VTS-VTSNAME (8 bytes)

This field returns the VTSNAME parameter from TBOPT or the VTS-NAME parameter from the VS command; the value from a VS command overrides the TBOPT value. For information on this parameter, see Appendix C on page 389.

5. FILLER (16 bytes)

This field is reserved for internal use.

6. VTS-TPVM (8 bytes)

Returns the TPVM name from TBOPT. For use when the tableBASE Process Manager option is used.

7. VTS-PREFIX (4 bytes)

This field returns the VTSPREFIX from TBOPT. It defines the prefix used for VTS-TRSs, when they are used in the ML library seach list. Default is "VTS:"; must be at least one character; minimum is ":".

VTS-NAME (8 bytes)

This parameter is used to pass the name of the VTS-TSR to the VS command. Names shorter than 8 characters must be blank-filled.

Note: With Release 6.0.3 and above, a VTS-TSR name can be up to 8 characters. Previous releases supported only 4-character VTS names. For compatibility with existing 4-character VTS names, if the last 4 characters of an 8-character VTS name are low-values (binary zeroes), they will be ignored when determining which VTS-TSR to access.

tableBASE parameters with C

```
typedef struct
{
   char  tbParmID[2];
   char  tbReserved1[2];
   char  tbVersion;
   char  tbFormat;
   char  tbReserved2[18];
   char tbSubSystem[8];
   char tbReserved3[4];
   char  tbTurboAnchor[8];
   char  tbReserved4[20];
} TbParmStruct;
```

```
typedef struct
{
   char  org;
   char  method;
   char  index;
   char  smc;
   char  readPassword[TB_PASSWORD_LENGTH];
   char  writePassword[TB_PASSWORD_LENGTH];
   int   rowSize;
   int   keySize;
   int   keyLocation;
   int   numberOfRows;
   short generations;
   short expansionFactor;
   short lowDensity;
   short highDensity;
   char  reserved1[6];
   char  dateTime[12];
   short absoluteGenerationNumber;
   char  datasetName[44];
   short relativeGenerationNumber;
   short generationsPresent;
   int   rowsAtExpand;
   char  DDName[8];
   char  baseTable[8];
   char  openStatus;
   char  altsInvoked;
   char  viewVersion;
   char  reserved2;
   char  userID[8];
   char  viewName[8];
   char  viewDate[12] ;
   char  userComments[16];
   char  vtsName[8];
   char  reserved3[68];
} TbTableDefinitionStruct;
```



```
typedef struct
{
   char tableName[TB_TABLE_NAME_LENGTH];
   char tableOpenStatus;
   char tableInOut;
   char tableAltInvoked;
   char tableCardinalId;
   int  filler;
   int  tableSize;
   int  tableRows;
   int  tableRwsBfExp;
   char tableTrueName[TB_BASE_TABLE_NAME_LENGTH];
   /* 44 BYTES FOR LT STAT UNTIL HERE */
   int  filler2;
   char tableTimeOpen[12];
   int  filler3;
   long long totalCallS;
   long long totalCallU;
   char vtsName 8 ;
} TbTableStatsStruct;
```

```
typedef struct
{
   int from;
   int reqd;
   int size;
   int total;
   int returned;
   int tsrHW;
   int openHW;
   int openNow;
   int tsrNow;
   int tsrSZ;
   int strobe;
   /* LT  EXTENDED FIELDS FOR V603 */
   int numOpenTable;
   long long tsrCallCont;
   int tsrMaxTable;
   int mapCmeSize;
   int tceHashSize;
   int tceIdxSize;
   int tceDefSize;
   int tsrAvailBlocks;
   char swSpaceManager;
   char filler[7];
} TbListBlockStruct;
```


5 tableBASE coding examples

The C and COBOL examples in this chapter illustrate the use of tableBASE commands. All parameters that appear in calls to the tableBASE API (TBLBASE) are described fully in Chapter 4. At the end of this chapter Assembler coding examples are also provided.

Summary of coding examples

The following commands are detailed in this chapter:

- Retrieval commands
  - Fetch rows by key
  - Fetch every row
  - Fetch every row from last to first
- Update commands
  - Insert a new row or replace it
  - Fetch a row and replace it
  - Empty a table
- Table control commands
  - Define a new table
  - Change the table name and definition in memory
  - Store a new generation
  - Copy a table from one tableBASE library to another
- Library maintenance commands
  - Delete all generations of a table
  - Rename a table
  - Initialize a tableBASE library
- System control commands

Coding conventions

Tables are often shared by multiple applications. COBOL copybooks for tableBASE structures should be entered in a common copy library and used where appropriate in all tableBASE applications.

Naming conventions should be established for all tableBASE tables and libraries. Every organization has specific requirements for naming conventions, but a few recommendations are offered here.

COBOL variable names

If a program uses more than one tableBASE table, the use of a different command area for each table is strongly recommended. This will allow the program to provide greater efficiency.

A typical COBOL structure describing a row area for the table HOS2RATE follows:

```
01  W-HOS2RATE-ROW-AREA.
   10  W-HOS2RATE-KEY-FIELDS.
       20  W-HOS2RATE-CUSTOMER-CLASS-CODE        PIC X(2).
       20  W-HOS2RATE-SPECIAL-DISC-CODE          PIC X(4).
   10  W-HOS2RATE-RATE                           PIC 999V99.
```


All COBOL variable names in parameters that relate to a specific table should be prefixed by the table name, as shown in the example above. In addition, it is recommended that prefixes include codes which denote the type of storage used for the variable (for example, W for working storage, L for the linkage section, T for the transaction work area, etc.).

Such variable names appear in the COBOL examples of parameter code described in Chapter 4 and in the sample code below. Identical code is provided in the distributed tableBASE Education dataset, in the member YOUR.PREFIX.TBASE.SRC(TBPARMS). Variable names that relate to a single table are prefixed by "xxxx" to indicate a coding template that requires modification. The generic "xxxx" is intended to be replaced by a specific table name when used in an application.

Programming in C with tableBASE

This section provides coding examples for various software languages.

Pragma linkage directive

The pragma linkage directive identifies TBLBASE as an external entry point (written in a different language).

```
#ifdef __cplusplus
   extern "OS" void TBLBASE( void * a, void * b, void * c, void * d );
#else
   #pragma linkage( TBLBASE, OS )
#endif
```

Strings

tableBASE does not use NULL-terminated strings for the different fields of type character [ ] (table names, passwords, etc.). If any of these strings are less than the maximum allowed size, they must be padded with spaces (the character 0x40 in EBCDIC).

An example of a routine to add this padding (as required) to a standard C NULL-terminated string is provided as part of the C examples in this manual. Its prototype is

```
fixStringLength( szInput, sOutput, int nMaximumLength );
```


Optional parameters

Some tableBASE commands allow for optional parameters to be passed to the tableBASE application programming interface (TBLBASE). It is very important that you do not pass NULL as the optional parameter to the call.

For example, GD (Get Definition) requires three parameters. The fourth parameter is the optional generation number. If the optional fourth parameter is supplied (even if NULL), tableBASE automatically looks in the library, directly bypassing the TSR, and incurs I/O. When the generation number is not supplied, TBLBASE checks the TSR for the opened table first before checking the library list. If found in the TSR, the results are returned, and any I/O is avoided. GD called with three parameters:

```
TBLBASE( tbParm, tbCommArea, tbDef);
```

As a second example, FK (Fetch by Key) requires three parameters. The fourth parameter is the optional Key Area. If the search key is in the Row Area and not in the Key Area, TBLBASE should be called with three parameters. Setting the fourth parameter to NULL might not cause the program to crash, but it is very likely you will never find the row, even if the row exists in the table.

There is another similar situation, in which the third parameter is not needed but the fourth parameter is. In this case, the solution is to use dummy parameters, variables of suitable type which are never used elsewhere in the program. They simply become placeholders in the parameters passed to TBLBASE.

Initializing the C data structures

This code segment is an example of how to initialize the C data structures for tableBASE before calling the tableBASE API (TBLBASE) with these data structures.

```
/*
 * This file contains examples of functions that can be used to
 * initialize the data structs needed for calling tableBASE.
 * The definitions of these structs are in dkh.h
 */


#include <stdio.h>
#include <string.h>
#include "dkh.h"

void InitTbParm( TbParmStruct * pTbP )
{
   /* TB-PARM */

   /*
    * Initialize it all to zero and set individual members as needed.
    */
   memset( pTbP, 0, sizeof(TbParmStruct) );

   pTbP->tbParmID[0] = 'T';
```



```
   pTbP->tbParmID[1] = 'B';
   pTbP->tbVersion   = '5';
   pTbP->tbFormat    = '0';
}

void InitTbCommandArea( TbCommandAreaStruct * pTbC, char * szTableName )
{
   int tableNameSize = 0, i;

   /* COMMAND-AREA */
   memset( pTbC->tbCommand, ' ', 2 );
   memset( pTbC->tbTable,   ' ', 8 );
   pTbC->tbFound         = ' ';
   pTbC->tbIndirectOpen  = 0;
   pTbC->tbReserved1     = 0;
   pTbC->tbAbendOverride = ' ';
   pTbC->tbError         = 0;
   pTbC->tbCount         = 0;
   memset( pTbC->tbLock, ' ', 8 );
   pTbC->tbRowOverrideLength = 0;
   pTbC->tbRowActualLength   = 0;
   pTbC->tbFgKeyLength       = 0;
   pTbC->tbFunctionID        = 0;
   memset( pTbC->tbDateArea, 0, 8 );
   memset( pTbC->tbReserved2, 0, 20 );/* 24 bytes for tableBASE v5.x */
   pTbC->tbReturnedAbsGenNo = 0;      /* added in tableBASE v6 */
   pTbC->tbErrorSubcode = 0;          /* added in tableBASE v6 */

   tableNameSize = strlen(szTableName);
   if( tableNameSize > 8 )
   {
      /*
       * Handle the error here, we truncate it for now.
       */
      strncpy( pTbC->tbTable, szTableName, 8 );
   }
   else
   {
      /* tableName length is less than 8, we append it with spaces */
      strncpy( pTbC->tbTable, szTableName, tableNameSize );
      for( i = tableNameSize; i < 8; i++ )
         pTbC->tbTable[i] = ' ';
   }
}

void InitTableDef( TbTableDefinitionStruct * pTbTD )
{
   /* DEFINITION-BLOCK */

   /*
    * Initialize it all to zero and set individual members as needed.
    */
   memset( pTbTD, 0, sizeof(TbTableDefinitionStruct) );

   pTbTD->org    = ' ';
   pTbTD->method = 'S';
   pTbTD->index  = 'T';
```

```
   pTbTD->smc    = 'R';
   memset( pTbTD->readPassword,  ' ', 8 );
   memset( pTbTD->writePassword, ' ', 8 );
   pTbTD->rowSize         = 120;
   pTbTD->keySize         = 11;
   pTbTD->keyLocation     = 1;
   pTbTD->numberOfRows    = 500;
   pTbTD->generations     = 1;
   pTbTD->expansionFactor = 250;
   pTbTD->lowDensity      = 500;
   pTbTD->highDensity     = 800;
   memset( pTbTD->dateTime, ' ', 12 );
   pTbTD->absoluteGenerationNumber = 10;
   memset( pTbTD->datasetName, ' ', 44 );
}

void InitLibList( TbLibListStruct * pTbL )
{
   memset( pTbL, ' ', 80 );
}

void InitTbLibSpace( TbLibSpaceStruct * pTbLS )
{
   pTbLS->libBlocksAllocated = 0;
   pTbLS->libBlocksRemaining = 0;
}

void InitTbDirSpec( TbDirSpecStruct * pTbDS )
{
   memset( pTbDS, 0, sizeof(TbDirSpecStruct) );
}

void InitTbAccDefStruct( TbAccDefStruct * pTbAD )
{
   /*
    * Initialize it all to zero and set individual members as needed.
    */
   memset( pTbAD, 0, sizeof(TbAccDefStruct) );

   pTbAD->numberOfRows = 0;
   pTbAD->rowSize      = 100;
   pTbAD->keySize      = 7;
   pTbAD->keyLocation  = 4;
   pTbAD->maxRows      = 1000;
   pTbAD->subscript    = 0;
   pTbAD->org          = 'S';
   pTbAD->action       = 'S';
   pTbAD->searchMethod = 'C';
}


void InitTbStats( TbTableStatsStruct * pTbTS, int nNumberOfTables )
{
   /*
    * The number of tables (nNumberOfTables) must be equal to
    * the value of the field TbListBlockStruct::reqd
    *
```



```
     * Do NOT use this function if you set TbListBlockStruct::size to
     * something different than sizeof(TbTableStatsStruct).
     */
    if ( nNumberOfTables > 0 )
       memset( pTbTS, 0, sizeof(TbTableStatsStruct)*nNumberOfTables );
    else
    {
       /*
        * Handle error code here, if necessary.
        */
    }
}

void InitListBlockStruct( TbListBlockStruct * pTbLB )
{
    /*
     * Initialize it all to zero and set individual members as needed.
     */
    memset( pTbLB, 0, sizeof(TbListBlockStruct) );

    pTbLB->from = 1;
    pTbLB->reqd = 96;

    /*
     * The amount of information returned (by the command LT)
     * for each table can be modified with the next field.
     * If set to something less than sizeof(TbTableStatsStruct),
     * the fourth parameter to TBLBASE() would be an array of
     * a somewhat truncated TbTableStatsStruct.
     * Use this with CAUTION.
     */
    pTbLB->size = sizeof(TbTableStatsStruct);
}

void InitLibraryAllocStruct( TbLibraryAllocStruct * pTbLA )
{
    pTbLA->libShareStatus = 'S';
    memset( pTbLA->libDataSetName, ' ', 44 );
}

void InitAltDefinitionStruct( TbAltDefinitionStruct * pTbAD )
{
    pTbAD->org         = ' ';
    pTbAD->method      = 'S';
    pTbAD->keyCount    = 1;
    pTbAD->keyLocation = 1;
    pTbAD->keySize     = 1;
}

int fixStringLength( char* input, char* output, int len )
{
    int size = 0, i = 0;

    size = strlen(input);
    if( size > len )
    {
       /*
```

```
         * Handle the error here, we truncate it for now.
         */
        strncpy( output, input, len );
    }
    else
    {
        /*
         * If tableName length is less than len, we append it
         * with spaces.
         */
        strncpy( output, input, size );
        for( i = size; i < len; i++ )
           output[i] = ' ';
    }
    return 0;
}
```

Retrieval commands

This section provides examples of COBOL retrieval commands.

Fetch rows by key

```
*
*    FETCH ROWS FROM A TABLE BASED ON KEY-AREA
*
     MOVE 'FK'                       TO xxxx-COMMAND.
     CALL 'TBLBASE' USING            TB-PARM
                                     xxxx-COMMAND-AREA
                                     xxxx-ROW-AREA
                                     xxxx-KEY-AREA.
     IF xxxx-FOUND = 'Y'
        PERFORM  FOUND-ROUTINE
     ELSE
        PERFORM  NOT-FOUND-ROUTINE.
```

Fetch every row

```
*
*    FETCH EVERY ROW FROM A TABLE
*
     MOVE 'GN'                       TO xxxx-COMMAND.
     MOVE ZERO                       TO xxxx-COUNT.
*
*    NOTE: THE FIRST GN CALL SETS THE FOUND CODE AS THE CONTROL FOR
*    THE LOOP.
*           (IT MAY BE AN EMPTY TABLE).
*
     PERFORM CALL-TBLBASE.
     PERFORM PROCESS-RETRIEVED-ROW UNTIL xxxx-FOUND = 'N'.
      ...
```



```
 CALL-TBLBASE.
     CALL 'TBLBASE' USING            TB-PARM
                                     xxxx-COMMAND-AREA
                                     xxxx-ROW-AREA.
 CALL-TBLBASE-EXIT.
     EXIT.
      ...
 PROCESS-RETRIEVED-ROW.
*    DO APPLICATION PROCESSING FOR THIS ROW, THEN GET ANOTHER
*    ROW...
      ...
     PERFORM CALL-TBLBASE.
 PROCESS-RETRIEVED-ROW-EXIT.
     EXIT.
```

Fetch every row from last to first

```
*
*    FETCH EVERY ROW IN A TABLE FROM THE LAST TO THE FIRST
*
     MOVE 'GL'                    TO xxxx-COMMAND.
*    NOTE: THE GL CALL SETS THE FOUND CODE AS THE CONTROL FOR THE
*    LOOP.
*           (IT MAY BE AN EMPTY TABLE).
*    THIS CODE USES THE SAME CALL-TBLBASE AND PROCESS-RETRIEVED-ROW
*    PARAGRAPHS AS THE PREVIOUS EXAMPLE.
     PERFORM CALL-TBLBASE.
     MOVE 'GP'                    TO xxxx-COMMAND.
     PERFORM PROCESS-RETRIEVED-ROW UNTIL xxxx-FOUND = 'N'.
```

Update commands

The following COBOL examples are appropriate for a single-task batch process where the program has exclusive control of the table. In an online process where several users may be updating different rows in the same table, RK commands should replace the RC commands.

Note: If multiple users could be updating a table—whether online, in a Read/Write VTS-TSR or other multi-tasking environment—be aware that a LOCK-LATCH password can be used to ensure exclusive access to a table.

Insert a new row or replace it

```
*
*     INSERT A NEW ROW OR REPLACE IT IN THE TABLE
*
      MOVE 'IK'                          TO xxxx-COMMAND.
      CALL 'TBLBASE' USING               TB-PARM
                                         xxxx-COMMAND-AREA
                                         xxxx-ROW-AREA.
      IF xxxx-FOUND = 'Y'
*        ROW FOUND SO NOT INSERTED.  REPLACED INSTEAD.
         MOVE 'RC'                       TO xxxx-COMMAND
         CALL 'TBLBASE' USING            TB-PARM
                                         xxxx-COMMAND-AREA
                                         xxxx-ROW-AREA.
```

Fetch a row and replace it

```
*
*     FETCH A ROW, UPDATE AND REPLACE IT IN THE TABLE
*     IF NOT FOUND, INSERT A NEW INITIALIZED ROW IN THE TABLE
*
      MOVE 'FK'                       TO xxxx-COMMAND.
      CALL 'TBLBASE' USING            TB-PARM
                                      xxxx-COMMAND-AREA
                                      xxxx-ROW-AREA
                                      xxxx-KEY-AREA.
      IF xxxx-FOUND = 'Y'
         MOVE 'RC'                    TO xxxx-COMMAND
      ELSE
         MOVE 'IC'                    TO xxxx-COMMAND.
      CALL 'TBLBASE' USING            TB-PARM
                                      xxxx-COMMAND-AREA
                                      xxxx-ROW-AREA.
```

Empty a table

```
*
*     REMOVE ALL ROWS FROM A TABLE
*
      MOVE 'OW'                       TO xxxx-COMMAND.
      CALL 'TBLBASE' USING            TB-PARM
                                      xxxx-COMMAND-AREA.
      MOVE 'MT'                       TO xxxx-COMMAND.
      CALL 'TBLBASE' USING            TB-PARM
                                      xxxx-COMMAND-AREA.
      MOVE 'ST'                       TO xxxx-COMMAND.
      CALL 'TBLBASE' USING            TB-PARM
                                      xxxx-COMMAND-AREA.
      MOVE 'CL'                       TO xxxx-COMMAND.
      CALL 'TBLBASE' USING            TB-PARM
                                      xxxx-COMMAND-AREA.
```

Table control commands

This section provides examples of table control commands written in COBOL.

Define a new table

```
*
*    DEFINE A NEW TABLE
*

     MOVE 'H'                    TO xxxx-ORG.
     MOVE 'H'                    TO xxxx-METHOD.
     MOVE 'DT'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 xxxx-DEFINITION-BLOCK.
```

Change the table name and definition in memory

```
*
*    CHANGE THE DEFINITION AND NAME OF AN EXISTING TABLE
*
     MOVE 'OR'                   TO xxxx-COMMAND.
     MOVE 'TABLE1'               TO xxxx-TABLE.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA.
     MOVE 'CN'                   TO xxxx-COMMAND.
     MOVE 'TABLE2'               TO NEW-TABLE-NAME.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 NEW-TABLE-NAME.
     MOVE 'TABLE2'               TO xxxx-TABLE.
     MOVE NEW-KEYSIZE            TO xxxx-KSZ.
     MOVE NEW-KEYLOC             TO xxxx-KLOC.
     MOVE 'CD'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 xxxx-DEFINITION-BLOCK.
```

Note: In single-thread batch, you may use the CN command on an open-read table, but in multi-user environments (e.g. CICS, multi-thread batch, VTS), the table must be opened for write.

Store a new generation

```
*
*    STORE A NEW GENERATION OF A TABLE AND DISPLAY THE NEW
*    GENERATION NUMBER
*
     MOVE 'OW'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 xxxx-GENERATION.
```



```
***END OF PROCESSING.  SAVE UPDATED TABLE.
 E-O-J.
     MOVE 'ST'                    TO xxxx-COMMAND.
     CALL 'TBLBASE' USING         TB-PARM
                                  xxxx-COMMAND-AREA.
     MOVE 'GD'                    TO xxxx-COMMAND.
     CALL 'TBLBASE' USING         TB-PARM
                                  xxxx-COMMAND-AREA
                                  xxxx-DEFINITION-BLOCK.
     DISPLAY 'GENERATION OF TABLE xxxx CREATED IS ' xxxx-ABS-GEN-NO.
     MOVE 'CL'                    TO xxxx-COMMAND.
     CALL 'TBLBASE' USING         TB-PARM
                                  xxxx-COMMAND-AREA.
```

Copy a table from one tableBASE library to another

```
*
*    DIVERT THE MOST RECENT GENERATION OF A TABLE FROM
*    ONE TABLEBASE LIBRARY TO ANOTHER
*

     MOVE 'OW'                    TO xxxx-COMMAND.
     CALL 'TBLBASE' USING         TB-PARM
                                  xxxx-COMMAND-AREA
                                  xxxx-PASSWORD
                                  xxxx-GENERATION.

     MOVE 'DV'                    TO xxxx-COMMAND.
     MOVE 'NEWLIB'                TO NEW-LIBNAME.

     CALL 'TBLBASE' USING         TB-PARM
                                  xxxx-COMMAND-AREA
                                  NEW-LIBNAME.

     MOVE 'ST'                    TO xxxx-COMMAND.
     CALL 'TBLBASE' USING         TB-PARM
                                  xxxx-COMMAND-AREA.

     MOVE 'CL'                    TO xxxx-COMMAND.
     CALL 'TBLBASE' USING         TB-PARM
                                  xxxx-COMMAND-AREA.
```

Library maintenance commands

The following are examples of Library Maintenance commands written in COBOL.

Delete all generations of a table

```
*
*    ELIMINATE ALL GENERATIONS OF A TABLE
*
     MOVE 'XT'                    TO xxxx-COMMAND.
     MOVE 'TABLE1'                TO xxxx-TABLE.
     CALL 'TBLBASE' USING         TB-PARM
```

```
                                    xxxx-COMMAND-AREA
                                    xxxx-PASSWORD.
```

Rename a table

```
*
*    RENAME ALL GENERATIONS OF A TABLE
*
     MOVE 'RN'                      TO xxxx-COMMAND.
     MOVE 'NEWNAME'                 TO NEW-TABLE-NAME.
     CALL 'TBLBASE' USING           TB-PARM
                                    xxxx-COMMAND-AREA
                                    NEW-TABLE-NAME.
                                    XXXX-PASSWORD.
```

Initialize a tableBASE library

```
*
*    INITIALIZE A DATASET TO BE A tableBASE LIBRARY
*
     MOVE 'DL'                      TO xxxx-COMMAND.
     MOVE 'TEMPLIB'                 TO DDNAME.
     CALL 'TBLBASE' USING           TB-PARM
                                    xxxx-COMMAND-AREA
                                    DDNAME.
```

System control commands

Below are examples of system control commands written in COBOL.

Locate a table from a list of libraries

```
*
*    LOCATE A TABLE FROM A LIST OF tableBASE LIBRARIES
*
     MOVE 'TESTLIB'                 TO LIB-LIST-1.
     MOVE 'PRODLIB'                 TO LIB-LIST-2.
     MOVE SPACES                    TO LIB-LIST-3.
     MOVE 'ML'                      TO xxxx-COMMAND.
     CALL 'TBLBASE' USING           TB-PARM
                                    xxxx-COMMAND-AREA
                                    LIB-LIST.

     MOVE 'PAYTABLE'                TO xxxx-TABLE.
     MOVE 'GD'                      TO xxxx-COMMAND.
     CALL 'TBLBASE' USING           TB-PARM
                                    xxxx-COMMAND-AREA
                                    xxxx-DEFINITION-BLOCK.
     IF  xxxx-FOUND = 'Y'
         PERFORM  FOUND-ROUTINE
     ELSE
         PERFORM  NOT-FOUND-ROUTINE.
```


Save and restore current status

```
*
*    A GENERAL PURPOSE SUBROUTINE TO SAVE AND RESTORE
*    LIB-LIST AND STATUS-SWITCHES.
*
     MOVE 'LL'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 LIB-LIST-SAVE.
     MOVE 'LS'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 STATUS-SWITCHES-SAVE.

     MOVE 'ML'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 LIB-LIST-NEW.
     MOVE 'CS'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 STATUS-SWITCHES-NEW.

*    (normal processing until end of routine)
     MOVE 'ML'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 LIB-LIST-SAVE.
     MOVE 'CS'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 STATUS-SWITCHES-SAVE.
```

Sample tableBASE applications

The following are samples of tableBASE applications.

Fetch a row from a table

In this example, the table is not updated and is not password protected, so an auto-open will be used (no open command will be issued). At the end of the job, the table will not be closed but will be released by the operating system.

The search key is taken from within a data record obtained during processing.

In COBOL

```
 PROCEDURE DIVISION.
***  FETCH A TABLE ROW USING ITS KEY.
     MOVE 'FK'                   TO xxxx-COMMAND.
     CALL 'TBLBASE' USING        TB-PARM
                                 xxxx-COMMAND-AREA
                                 xxxx-ROW-AREA
                                 DATA-KEY.
     IF  xxxx-FOUND = 'Y'
         PERFORM  FOUND-ROUTINE
     ELSE
         PERFORM  NOT-FOUND-ROUTINE.
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX0C
 *
 *  Fetch a row from a table
 *
 *  In this example, we will fetch one row from
 *  the table with the key supplied by the user.
 *  Auto-open will be used and the table will be
 *  released upon program termination.
 *
 *  Please refer to other document for the
 *  initXXXX and fixStringLenght functions.
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szKey[6] = "TIGER";
static char szStatus[6] = "NYYYN";
```

```
static int nGen = 0;

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbTableDefinitionStruct tbTableDef;
   char * pRowArea = NULL;
   char * pKeyArea = NULL;
   char sStatus[8];
   char sTableName[8];
   int nGeneration = nGen;

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );
   InitTableDef( &tbTableDef );

   /*
    * Initialize tableBASE with CS, ChangeStatus.
    */
   fixStringLength( szStatus, sStatus, 8 );
   memcpy( tbCommArea.tbCommand, "CS", 2 );
   TBLBASE( &tbParm, &tbCommArea, sStatus );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "CS\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }

   /*
    * GD, get table definition to retrieve the row length and
    * key length.
    */
   memcpy( tbCommArea.tbCommand, "GD", 2 );
   TBLBASE( &tbParm, &tbCommArea, &tbTableDef, nGeneration );
   if( (tbCommArea.tbError != TB_SUCCESS) || tbCommArea.tbFound == 'N' )
   {
      printf( "GD\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }

   /*
    * Allocate spaces for a row (with an additional string
    * terminator) and a key.
    */
   pRowArea = (char *) malloc( tbTableDef.rowSize + 1 );
   if( pRowArea == NULL )
```



```
      return TB_ERROR;
   memset( pRowArea, ' ', tbTableDef.rowSize);

   pKeyArea = (char *) malloc( tbTableDef.keySize );
   if( pKeyArea == NULL )
      return TB_ERROR;
   memset( pKeyArea, ' ', tbTableDef.keySize);

   /*
    * Call tableBASE with FK, FetchbyKey.
    */
   fixStringLength( szKey, pKeyArea, tbTableDef.keySize );
   memcpy( tbCommArea.tbCommand, "FK", 2 );
   TBLBASE( &tbParm, &tbCommArea, pRowArea, pKeyArea );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "FK\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      free( pRowArea );
      free( pKeyArea );
      return tbCommArea.tbError;
   }
   else
   {
      /*
       * If the key is not found, it will print a row with spaces.
       * You can check the found code in CommandArea to make sure
       * the row was found.
       */
      pRowArea[tbTableDef.rowSize] = '\0';
      printf( "FK: %s\n", pRowArea );
   }
   if( pRowArea != NULL )
      free( pRowArea );
   if( pKeyArea != NULL )
      free( pKeyArea );
   return TB_SUCCESS;
}
```

Update an existing table

In this example, rows will be inserted by key. If the insert is successful, no further action is required. If the row is already in the table, the insert will not be done. Instead, the row will be replaced. To save another search, the replace can be done using the COUNT value set by the insert command. At the end of processing, the table will be stored and the generation number printed.

Note: If multiple users could be updating a table—whether online, in a Read/Write VTS-TSR or other multitasking environment—be aware that a LOCK-LATCH password can be used to ensure exclusive access to a table.

In COBOL

```
       PROCEDURE DIVISION.
       HOUSE-KEEPING.
           MOVE 'OW' TO xxxx-COMMAND
           CALL 'TBLBASE' USING TB-PARM
                                xxxx-COMMAND-AREA
                                xxxx-PASSWORD
                                GENERATION

           PERFORM WHILE THERE-IS-MORE-INPUT-DATA
      * (obtain data for next row)
               MOVE DATA-AREA TO xxxx-ROW-FIELDS
               MOVE DATA-KEY TO xxxx-ROW-KEY
      *** INSERT A ROW USING ITS KEY.
               MOVE 'IK' TO xxxx-COMMAND
               CALL 'TBLBASE' USING TB-PARM
                                    xxxx-COMMAND-AREA
                                    xxxx-ROW-AREA
               IF xxxx-FOUND = 'N'
      *** ROW FOUND SO NOT INSERTED. REPLACE INSTEAD.
                  MOVE 'RC' TO xxxx-COMMAND
                  CALL 'TBLBASE' USING TB-PARM
                                       xxxx-COMMAND-AREA
                                       xxxx-TABLE-ROW
                  END-IF
           END-PERFORM
           .

       E-O-J.
      *** END OF PROCESSING. SAVE UPDATED TABLE.
           MOVE 'ST' TO xxxx-COMMAND
           CALL 'TBLBASE' USING TB-PARM
                                xxxx-COMMAND-AREA
      *** DETERMINE CURRENT TABLE CHARACTERISTICS.
           MOVE 'GD' TO xxxx-COMMAND
           CALL 'TBLBASE' USING TB-PARM
                                xxxx-COMMAND-AREA
                                xxxx-DEFINITION-BLOCK
           DISPLAY 'GENERATION OF TABLE1 CREATED IS ' xxxx-ABS-GEN-NO
      *** CLOSE THE TABLE.
```



```
            MOVE 'CL' TO xxxx-COMMAND.
            CALL 'TBLBASE' USING TB-PARM
                                 xxxx-COMMAND-AREA
            .
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX1C
 *
 *  Update an existing table
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szStatus[6] = "NYYYN";
static char szWritePassword[2] = " ";
static int nGen = 0;
static char szKey[4] = "Pat";
static char szRow[21] = "Pat  123456789012345";

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbTableDefinitionStruct tbTableDef;
   char sWritePassword[8];
   char sStatus[8];
   char sTableName[8];
   int nGeneration = nGen;
   char * pKeyArea = NULL;
   char * pRowArea = NULL;

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );
   InitTableDef( &tbTableDef );

   /*
    * Initialize tableBASE with CS, ChangeStatus.
    */
   fixStringLength( szStatus, sStatus, 8 );
   memcpy( tbCommArea.tbCommand, "CS", 2 );
   TBLBASE( &tbParm, &tbCommArea, sStatus );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
```

```
   printf( "CS\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Call tableBASE with OW, OpenforWrite.
 */
fixStringLength( szWritePassword, sWritePassword, 8 );
memcpy( tbCommArea.tbCommand, "OW", 2 );
TBLBASE( &tbParm, &tbCommArea, sWritePassword, nGeneration);
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "OW\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * GD, get table definition to retrieve the row length
 * and the key length.
 */
memcpy( tbCommArea.tbCommand, "GD", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDef, nGeneration );
if( (tbCommArea.tbError != TB_SUCCESS)
      || (tbCommArea.tbFound == 'N') )
{
   printf( "GD\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Allocate space for a row and a key.
 */
pRowArea = (char *) malloc( tbTableDef.rowSize );
if( pRowArea == NULL )
   return TB_ERROR;
memset( pRowArea, ' ', tbTableDef.rowSize);

pKeyArea = (char *) malloc( tbTableDef.keySize );
if( pKeyArea == NULL ) {
   free( pRowArea );
   return TB_ERROR;
}
memset( pKeyArea, ' ', tbTableDef.keySize );

fixStringLength( szKey, pKeyArea, tbTableDef.keySize );
fixStringLength( szRow, pRowArea, tbTableDef.rowSize );

/*
```



```
 * Call tableBASE with IK, InsertByKey.
 */
memcpy( tbCommArea.tbCommand, "IK", 2 );
TBLBASE( &tbParm, &tbCommArea, pRowArea, pKeyArea);
if( tbCommArea.tbFound == 'Y' )
{
   /*
    * Call tableBASE with RC, ReplacebyCount.
    */
   memcpy( tbCommArea.tbCommand, "RC", 2 );
   TBLBASE( &tbParm, &tbCommArea, pRowArea);
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "RC\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      free( pRowArea );
      free( pKeyArea );
      return tbCommArea.tbError;
   }
}

/*
 * Call tableBASE with ST, StoreTable.
 */
memcpy( tbCommArea.tbCommand, "ST", 2 );
TBLBASE( &tbParm, &tbCommArea );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "ST\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   free( pRowArea );
   free( pKeyArea );
   return tbCommArea.tbError;
}

memcpy( tbCommArea.tbCommand, "GD", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDef, nGeneration );
if( (tbCommArea.tbError != TB_SUCCESS)
      || (tbCommArea.tbFound == 'N') )
{
   printf( "GD\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   free( pRowArea );
   free( pKeyArea );
   return tbCommArea.tbError;
}
else
{
   printf( "Generation of %s table: %d\n",
            szTableName, tbTableDef.generations );
}
```



```
   /*
    * Call tableBASE with CL, CLoseTable.
    */
   memcpy( tbCommArea.tbCommand, "CL", 2 );
   TBLBASE( &tbParm, &tbCommArea );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "CL\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      free( pRowArea );
      free( pKeyArea );
      return tbCommArea.tbError;
   }

   if( pRowArea != NULL )
      free( pRowArea );
   if( pKeyArea != NULL )
      free( pKeyArea );

   return TB_SUCCESS;
}
```

Sequentially process all rows in a table

To improve program readability and facilitate debugging, the table will be opened with an explicit open command. To obtain the rows in a sequence, the FC (Fetch by Count) command will be used with full manual control of the counter.

In COBOL

```
 PROCEDURE DIVISION.
 HOUSE-KEEPING.
```

```
        PROCEDURE DIVISION.
        HOUSE-KEEPING.
       *** OPEN A TABLE FOR WRITE AND SET COMMAND TO FETCH BY COUNT.
            MOVE 'OW' TO xxxx-COMMAND
            CALL 'TBLBASE' USING TB-PARM
                                 xxxx-COMMAND-AREA
                                 xxxx-PASSWORD
                                 GENERATION
            MOVE 'FC' TO xxxx-COMMAND
            MOVE ZERO TO xxxx-COUNT
            MOVE ' '  TO xxxx-FOUND
            PERFORM UNTIL  xxxx-FOUND = 'N'
                ADD +1 TO xxxx-COUNT
                CALL 'TBLBASE' USING TB-PARM
                                     xxxx-COMMAND-AREA
                                     xxxx-ROW-AREA
```



```
                IF xxxx-FOUND = 'Y'
        *           (process table row obtained)
                END-IF
         END-PERFORM
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"
/*
 *  DK1TEX2C
 *
 *  Process every row in a table sequentially.
 *
 *  This program will explicitly open the table for write,
 *  So no other program can change the data while the program
 *  is reading the table.
 *  But we did not explicitly close the table after the fetch
 *  by count. It is because once the batch job is finish execution.
 *  The operation system will release the table.
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szStatus[6] = "NYYYN";
static char szWritePassword[2] = " ";
static int nGen = 0;

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbTableDefinitionStruct tbTableDef;
   char sWritePassword[8];
   char * pRowArea = NULL;
   char sStatus[8];
   char sTableName[8];
   int nGeneration = nGen;
   int nCount = 0;
   int notFound = 1;

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );
   InitTableDef( &tbTableDef );

   /*
    * Initialize tableBASE with CS, ChangeStatus.
```

```
 */
fixStringLength( szStatus, sStatus, 8 );
memcpy( tbCommArea.tbCommand, "CS", 2 );
TBLBASE( &tbParm, &tbCommArea, sStatus );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "CS\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Call tableBASE with OW, OpenforWrite.
 */
fixStringLength( szWritePassword, sWritePassword, 8 );
memcpy( tbCommArea.tbCommand, "OW", 2 );
TBLBASE( &tbParm, &tbCommArea, sWritePassword, nGeneration);
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "OW\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * GD, get table definition to retrieve both the row length
 * and key length.
 */
memcpy( tbCommArea.tbCommand, "GD", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDef, nGeneration );
if( (tbCommArea.tbError != TB_SUCCESS)
      || (tbCommArea.tbFound == 'N') )
{
   printf( "GD\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Allocate space for a row (with an additional string
 * terminator).
 */
pRowArea = (char *) malloc( tbTableDef.rowSize + 1 );
if( pRowArea == NULL )
   return TB_ERROR;
memset( pRowArea, ' ', tbTableDef.rowSize);

memcpy( tbCommArea.tbCommand, "FC", 2 );
nCount = 0;
while( notFound )
{
```

```
      tbCommArea.tbCount = ++nCount;
      /* Call tableBASE with FC, FetchbyCount */
      TBLBASE( &tbParm, &tbCommArea, pRowArea );
      if( tbCommArea.tbFound == 'N' )
         notFound = 0;
      else
      {
         pRowArea[tbTableDef.rowSize] = '\0';
         printf( "Row %d: %s\n", nCount, pRowArea );
         notFound = 1;
      }
   }

   if( pRowArea != NULL )
      free( pRowArea );

   return TB_SUCCESS;
}
```

Generic search

Search a table to find the group of records that begin with the same truncated version of the table key. Change the table organization to sequential if presently random, user-ordered or hash; this is necessary because the command Fetch Generic (FG) operates only on sequentially organized tables. Access the table using a generic key by setting the low order part of key to the delimiter (usually an asterisk *). The first FG command will retrieve the first row in the group, and subsequent calls will retrieve the rest of the group. The user must test the found code after each call to make sure the group or table has not been exhausted. For more information see "Fetch Generic (FG)" on page 91.

In COBOL

```
       01 xxxx-SEARCH-KEY.
           05 xxxx-SEARCH-KEY-1     PIC X(04) VALUE 'aaaa'.
           05 xxxx-SEARCH-KEY-2     PIC X VALUE '*'.
           05 xxxx-SEARCH-KEY-3     PIC X(04).

       PROCEDURE DIVISION.

      *** SETUP SEQUENTIAL/BINARY SEARCH.
      *** CHANGE-ORG.
           MOVE LOW-VALUES TO xxxx-DEFINITION-BLOCK
           MOVE 'S' TO xxxx-ORG
           MOVE 'B' TO xxxx-METHOD
           MOVE 'CD' TO xxxx-COMMAND
           CALL 'TBLBASE' USING TB-PARM
                                xxxx-COMMAND-AREA
                                xxxx-DEFINE
      *** SET UP GENERIC KEY
           MOVE PART-NO-1 TO xxxx-SEARCH-KEY-1
           MOVE ZERO TO xxxx-COUNT
           MOVE 'FG' TO xxxx-COMMAND
```



```
           MOVE 'Y'  TO xxxx-FOUND
           PERFORM  WHILE   xxxx-FOUND = 'Y'
               CALL 'TBLBASE' USING TB-PARM
                                    xxxx-COMMAND-AREA
                                    xxxx-ROW-AREA
                                    xxxx-SEARCH-KEY
               IF xxxx-FOUND = 'Y'
      ***          (process table row)
               END-IF
           END-PERFORM
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX3C
 *
 *  This program will fetch all rows which satisfy the generic
 *  search condition (in this case, that their key starts with
 *  the string "P").
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szStatus[6] = "NYYYN";
static char szWritePassword[2] = " ";
static char szSearchKey[2] = "P";
static int nGen = 0;
static int nKeyLength = 1;

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbTableDefinitionStruct tbTableDef;
   char * pRowArea = NULL;
   char * pSearchKey = NULL;
   char sStatus[8];
   char sTableName[8];
   int nGeneration = nGen;
   int notFound = 1;

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );
```



```
InitTableDef( &tbTableDef );

/*
 * Initialize tableBASE with CS, ChangeStatus.
 */
fixStringLength( szStatus, sStatus, 8 );
memcpy( tbCommArea.tbCommand, "CS", 2 );
TBLBASE( &tbParm, &tbCommArea, sStatus );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "CS\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Setup Sequential/Binary Search with CD, ChangeDefinition.
 */
memcpy( tbCommArea.tbCommand, "CD", 2 );
tbTableDef.org = 'S';
tbTableDef.method = 'B';
TBLBASE( &tbParm, &tbCommArea, &tbTableDef, nGeneration );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "CD\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * GD, get table definition to retrieve the row length.
 */
memcpy( tbCommArea.tbCommand, "GD", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbTableDef, nGeneration );
if( (tbCommArea.tbError != TB_SUCCESS)
      || (tbCommArea.tbFound == 'N') )
{
   printf( "GD\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Allocate space for a row (with an additional string
 * terminator) and for the key.
 */
pRowArea = (char *) malloc( tbTableDef.rowSize + 1 );
if( pRowArea == NULL )
   return TB_ERROR;
memset( pRowArea, ' ', tbTableDef.rowSize);
pSearchKey = (char *) malloc( tbTableDef.keySize );
```



```
   if( pSearchKey == NULL )
   {
      free( pRowArea );
      return TB_ERROR;
   }
   memset( pSearchKey, ' ', tbTableDef.keySize );

   /*
    * Call tableBASE with FG, FetchGeneric.
    */
   memcpy( tbCommArea.tbCommand, "FG", 2 );
   tbCommArea.tbCount = 0;
   tbCommArea.tbFgKeyLength = nKeyLength;
   fixStringLength( szSearchKey, pSearchKey, tbTableDef.keySize );
   while( notFound )
   {
      TBLBASE( &tbParm, &tbCommArea, pRowArea, pSearchKey );
      if( tbCommArea.tbFound == 'Y' ) {
         pRowArea[tbTableDef.rowSize] = '\0';
         printf( "Found at Row %d: %s\n",
                  tbCommArea.tbCount, pRowArea );
         notFound = 1;
      }
      else
      {
         notFound = 0;
      }
   }
   /*
    * Call tableBASE with CL, CloseTable.
    */
   memcpy( tbCommArea.tbCommand, "CL", 2 );
   TBLBASE( &tbParm, &tbCommArea );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "CL\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      free( pRowArea );
      free( pSearchKey );
      return tbCommArea.tbError;
   }
   if( pRowArea != NULL )
      free( pRowArea );
   if( pSearchKey != NULL )
      free( pSearchKey );

   return TB_SUCCESS;
}
```

Define a dynamic table to summarize data in memory

Use a hash organization and search method for maximum efficiency when performing random keyed access on a table. A summary table will be used here to accumulate the total value for an unknown number of parts. At the end of the job, the totals are to be printed in key sequence. Once a table row is found, the table need not be searched again; it can be referenced directly with any of the COUNT commands, such as Replace by Count.

At end of file for the input data, the table is sorted by changing the definition to sequential and the part number and total for each part are printed.

Note: If multiple users could be updating a table—whether online, in a Read/Write VTS-TSR or other multitasking environment—be aware that a LOCK-LATCH password is required to ensure exclusive access to a table.

In COBOL

```
DATA DIVISION.
01  DATA-ROW.
    05
    05  PARTNO                          PIC X(8).
    05
    05  PART-VALUE                      PIC S9(5)    COMP-3.
    05

01  SUM-ROW-AREA.
    05  SUM-ROW-PARTNO-KEY              PIC X(8).
    05  SUM-ROW-ACCUM-VALUE             PIC S9(11) COMP-3.

01  SUM-COMMAND-AREA.
    05  SUM-COMMAND                     PIC XX       VALUE SPACES.
    05  SUM-TABLE                       PIC X(8)     VALUE 'TABLE1'.
    05  SUM-FOUND                       PIC X        VALUE SPACES.
    05  SUM-INDIRECT-OPEN               PIC X        VALUE LOW VALUES.
    05  SUM-RESERVED                    PIC X        VALUE LOW-VALUES.
    05  SUM-ABEND-OVERRIDE              PIC X        VALUE SPACES.
    05  SUM-ERROR-CODE                  PIC S9(4)    COMP VALUE +0.
    05  SUM-COUNT                       PIC S9(9)    COMP VALUE +0.
    05  SUM-LOCK-LATCH                  PIC X(8)     VALUE SPACES.
    05  SUM-LENGTHS.
        10  SUM-ROW-OVERRIDE-LENGTH     PIC S9(9)    COMP VALUE +0.
        10  SUM-ROW-ACTUAL-LENGTH       PIC S9(9)    COMP VALUE +0.
        10  SUM-FG-KEY-LENGTH           PIC S9(4)    COMP VALUE +0.
    05  SUM-FUNCTION-ID                 PIC S9(4)    COMP VALUE +0.
    05  SUM-FUNCTION-AREA               PIC X(28)    VALUE LOW-VALUES.
    05  SUM-DATE-AREA                   REDEFINES SUM-FUNCTION-AREA.
        10  SUM-DATE                    PIC X(8).
        10  RESERVED                    PIC X(20).
     05  SUM-RETURNED-ABS-GEN-NO        PIC S9(4)    COMP VALUE +0.
    05  SUM-ERROR-SUBCODE               PIC S9(4)    COMP VALUE +0.
```

```
01  SUM-DEFINITION-BLOCK.
    05  SUM-ORG                          PIC X.
    05  SUM-METHOD                       PIC X.
    05  SUM-INDEX                        PIC X        VALUE 'P'.
    05  SUM-SMC                          PIC X        VALUE SPACES.
    05  SUM-RPSWD                        PIC X(8)     VALUE SPACES.
    05  SUM-WPSWD                        PIC X(8)     VALUE SPACES.
    05  SUM-RSZ                          PIC S9(9)    COMP VALUE +14.
    05  SUM-KSZ                          PIC S9(9)    COMP VALUE +8.
    05  SUM-KLOC                         PIC S9(9)    COMP VALUE +1.
    05  SUM-ROWS                         PIC S9(9)    COMP VALUE +500.
    05  SUM-GENERATIONS                  PIC S9(4)    COMP VALUE +1.
    05  SUM-EXP-FACT                     PIC S9(4)    COMP VALUE +200.
    05  SUM-LO-DEN                       PIC S9(4)    COMP VALUE +500.
    05  SUM-HI-DEN                       PIC S9(4)    COMP VALUE +800.
    05  FILLER                           PIC X(6).
    05  SUM-DATE-TIME                    PIC X(12)    VALUE SPACES.
    05  SUM-ABS-GEN-NO                   PIC S9(4)    COMP VALUE +0.
    05  SUM-DATASET-NAME                 PIC X(44).
    05  SUM-REL-GEN-NO                   PIC S9(4)    COMP.
   05  SUM-GENS-PRESENT                  PIC S9(4)    COMP.
   05  SUM-ROWS-AT-EXPAND                PIC S9(9)    COMP.
   05  SUM-DDNAME                        PIC X(8).
   05  SUM-DATA-TABLE                    PIC X(8).
   05  SUM-OPEN-STATUS                   PIC X.
   05  SUM-ALTS-INVOKED                  PIC X.
   05  SUM-VIEW-VERSION                  PIC X.
   05  FILLER                            PIC X.
   05  SUM-USERID                        PIC X(8).
   05  SUM-VIEW-NAME                     PIC X(8).
   05  SUM-VIEW-DATE                     PIC X(12).
   05  SUM-USER-COMMENTS                 PIC X(16).
   05  SUM-VTSNAME                       PIC X(8)     VALUE SPACES.
   05  FILLER                            PIC X(68).

     PROCEDURE DIVISION.
     PROCESS-ROUTINE.
         PERFORM DEFINE-TABLE
         PERFORM POPULATE-TABLE
         PERFORM CHANGE-TO-SEQUNETIAL-ORDER
         PERFORM PRINT-TABLE
         PERFORM CLOSE-TABLE
         GOBACK.

     DEFINE-TABLE
         MOVE 'H' TO SUM-ORG
         MOVE 'H' TO SUM-METHOD
         MOVE 'DT' TO SUM-COMMAND
         CALL 'TBLBASE' USING TB-PARM
                              SUM-COMMAND-AREA
                              SUM-DEFINITION-BLOCK
         EXIT.

     POPULATE-TABLE.
         PERFORM OBTAIN-INPUT-DATA-RECORD
         PERFORM UNTIL  (no more data records to be processed)
             MOVE PARTNO TO SUM-ROW-PARTNO-KEY
```



```
*** SEARCH THE TABLE FOR MATCHING PARTNO.
         MOVE 'FK' TO SUM-COMMAND
         CALL 'TBLBASE' USING TB-PARM
                              SUM-COMMAND-AREA
                              SUM-ROW-AREA
                              SUM-ROW-PARTNO-KEY
         IF SUM-FOUND = 'Y'
*** MATCHING PARTNO FOUND. UPDATE AND REPLACE TABLE ROW
             ADD PART-VALUE TO SUM-ROW-ACCUM-VALUE
             MOVE 'RC' TO SUM-COMMAND
             CALL 'TBLBASE' USING TB-PARM
                                  SUM-COMMAND-AREA
                                  SUM-ROW-AREA
         ELSE
*** MATCHING PARTNO NOT FOUND. INSERT NEW TABLE ROW
             MOVE PART-VALUE TO SUM-ROW-ACCUM-VALUE
             MOVE 'IC' TO SUM-COMMAND
             CALL 'TBLBASE' USING TB-PARM
                                  SUM-COMMAND-AREA
                                  SUM-ROW-AREA
         END-IF
         PERFORM OBTAIN-INPUT-DATA-RECORD
     END-PERFORM
     EXIT.

*** RE-SEQUENCE THE TABLE FOR PRINTING.
 CHANGE-TO-SEQUENTIAL-ORDER.
     MOVE 'S' TO SUM-ORG
     MOVE 'B' TO SUM-METHOD
     MOVE 'CD' TO SUM-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          SUM-COMMAND-AREA
                          SUM-DEFINITION-BLOCK
     EXIT.

*** PRINT ALL ROWS IN THE TABLE.
 PRINT-TABLE.
     MOVE 'GF' TO SUM-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          SUM-COMMAND-AREA
                          SUM-ROW-AREA
     MOVE 'GN' TO SUM-COMMAND
     PERFORM  UNTIL  SUM-FOUND =  'N'
         DISPLAY 'PART NUMBER ' SUM-ROW-PARTNO-KEY
                 ' QUANTITY - ' SUM-ROW-ACCUM-VALUE
         CALL 'TBLBASE' USING TB-PARM
                              SUM-COMMAND-AREA
                              SUM-ROW-AREA
     END-PERFORM
     EXIT.

 CLOSE-TABLE.
     MOVE 'CL' TO SUM-COMMAND
     CALL 'TBLBASE' USING TB-PARM
```

Access a table that may not exist

In this example, the name of a table is obtained from an input source. The program must open the table for write, from a library, or if that fails, the program must create a new table. To prevent an abend when opening a non-existent table, the abend status will be temporarily overridden to bypass abend processing. As soon as the open command has been issued, the status will automatically be reset to permit abends. If the table cannot be found on the library (error code 0009), a new table will be defined. If any other error occurs, repeating the open command will cause tableBASE to abend.

In COBOL

```
*** (table name is obtained from the data)
     MOVE TABLE-NAME TO xxxx-TABLE
*** FOR THE NEXT COMMAND TURN OFF TABLEBASE ABEND FUNCTION
     MOVE 'Y' TO xxxx-ABEND-OVERRIDE

*** TRY TO FIND TABLE
     MOVE 'OW' TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA

     IF xxxx-ERROR-CODE NOT = 0
         IF xxxx-ERROR-CODE = 9
*** TABLE NOT FOUND, DEFINE A NEW TABLE
          MOVE 'DT' TO xxxx-COMMAND
          CALL 'TBLBASE' USING TB-PARM
                               xxxx-COMMAND-AREA
                               xxxx-DEFINE-BLOCK
          ELSE
***  FOR ANY OTHER ERROR, CAUSE TABLEBASE TO ABEND
              MOVE 'OW' TO xxxx-COMMAND
              CALL 'TBLBASE' USING TB-PARM
                                   xxxx-COMMAND-AREA
          END-IF
     END-IF

*** (continue processing with open table)
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX5C
 *
 *  Access a table that may or may not exist.
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[7] = "AARON2";
static char szStatus[6] = "NYYYN";
static char szStatusAbend[6] = "YYYYN";
static char szWritePassword[2] = " ";
static int nGen = 0;
static int nRowSize = 20;
static int nKeySize = 5;
static int nKeyLocaiton = 1;

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbTableDefinitionStruct tbTableDef;
   char sWritePassword[8];
   char sStatus[8];
   char sTableName[8];
   int nGeneration = nGen;

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );
   /*
    * Initialize tableBASE with CS, ChangeStatus.
    * As usual, this disable the abend.
    */
   fixStringLength( szStatus, sStatus, 8 );
   memcpy( tbCommArea.tbCommand, "CS", 2 );
   TBLBASE( &tbParm, &tbCommArea, sStatus );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "CS\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }
   /*
    * Call tableBASE with OW, OpenforWrite.
```

```
 */
fixStringLength( szWritePassword, sWritePassword, 8 );
memcpy( tbCommArea.tbCommand, "OW", 2 );
TBLBASE( &tbParm, &tbCommArea, sWritePassword, nGeneration);
if( tbCommArea.tbError == 9 )
{
   /*
    * Create a new table with DT, DefineTable.
    */
   memcpy( tbCommArea.tbCommand, "DT", 2 );
   InitTableDef( &tbTableDef );
   memcpy( tbTableDef.writePassword, sWritePassword, 8 );
   tbTableDef.org = 'H';
   tbTableDef.method = 'H';
   tbTableDef.rowSize = nRowSize;
   tbTableDef.keySize = nKeySize;
   tbTableDef.keyLocation = nKeyLocaiton;
   TBLBASE( &tbParm, &tbCommArea, &tbTableDef );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "DT\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }
   else
   {
      printf( "A new table was created in memory.\n" );
   }
}
if( (tbCommArea.tbError != TB_SUCCESS)
      && (tbCommArea.tbError != 9) )
{
   /*
    * Any error from OW, will cause tableBASE to abend
    * (abort) if the abend switch is turned on.
    */
   /*
    * Setup tableBASE to enable abend on errors (the first
    * byte of the status is set to 'Y' instead of 'N').
    * A simpler way to do that would have been to set
    * tbCommArea.tbAbendOverride to 'Y' before calling
    * Open for Write (setting the tbAbendOverride field only
    * works once, it is automatically reset to ' ' by TBLBASE()).
    */
   fixStringLength( szStatusAbend, sStatus, 8 );
   memcpy( tbCommArea.tbCommand, "CS", 2 );
   TBLBASE( &tbParm, &tbCommArea, sStatus );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "CS\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }
```



```
        fixStringLength( szWritePassword, sWritePassword, 8 );
        memcpy( tbCommArea.tbCommand, "OW", 2 );
        TBLBASE( &tbParm, &tbCommArea, sWritePassword, nGeneration);
        /*
         * We should never reach this point.
         */
    }
    /*
     * You can do whatever you want with the newly defined table
     * or the table just opened. However, you should remember to
     * store and close the table before exiting the program,
     * otherwise all the changes to the table will be lost.
     */
    printf( "The table will be closed and released upon termination.\n" );

    return TB_SUCCESS;
}
```

Establish another key with which to search a table

A part table is usually searched with a part number as the key. To enter order information when the part number is missing from the order, the part name may be used as an alternate key for searching the table.

PARTNO is a table on the tableBASE library and its key is Part Number. The PARTNO table is opened and an Alternate Index called PARTNAME is Invoked with the key of Part Name. This now allows access to either the PARTNAME table or the PARTNO table, two routes to the same data.

Note: The Invoke Alternate (IA) command has been largely superseded by Open commands (OR and OW) that are capable of opening multiple Alternate Indexes for a single Data Table. However, the IA command remains the only way to generate a transient Alternate Index.

In COBOL

```
 DATA DIVISION.
 01 PART-ROW.
     05  PART-NUMBER                          PIC X(10).
     05  PART-NAME                            PIC X(25).
     05  PART-FIELDS...
 01  ORDER-PART-NAME                          PIC X(25) VALUE SPACES.
 01  ORDER-PART-NUMBER                        PIC X(10) VALUE SPACES.
 01  PARTNO-COMMAND-AREA.
     05  PARTNO-COMMAND                       PIC XX VALUE SPACES.
     05  PARTNO-TABLE                         PIC X(8) VALUE 'PARTNO'.
     05  PARTNO-FOUND                         PIC X VALUE SPACES.
     05  PARTNO-INDIRECT-IND                  PIC X VALUE LOW-VALUES.
     05  PARTNO-RESERVED                      PIC X VALUE LOW-VALUES.
     05  PARTNO-ABEND-OVERRIDE                PIC X VALUE SPACES.
     05  PARTNO-ERROR-CODE                    PIC S9(4) COMP VALUE +0.
     05  PARTNO-COUNT                         PIC S9(9) COMP VALUE +0.
     05  PARTNO-LOCK                          PIC X(8) VALUE SPACES.
```



```
     05  PARTNO-LENGTHS.
         10 PARTNO-ROW-OVERRIDE-LENGTH          PIC S9(9) COMP VALUE +0.
         10 PARTNO-ROW-ACTUAL-LENGTH            PIC S9(9) COMP VALUE +0.
         10 PARTNO-FG-KEY-LENGTH                PIC S9(4) COMP VALUE +0.
     05  PARTNO-FUNCTION-ID                     PIC S9(4) COMP VALUE +0.
     05  PARTNO-FUNCTION-AREA                   PIC X(8) VALUE LOW-VALUES.
     05  PARTNO-DATE-AREA            REDEFINES PARTNO-FUNCTION-AREA.
         10 PARTNO-DATE                         PIC X(8).
         10 RESERVED                            PIC X(20).
     05  PARTNO-RETURNED-ABS-GEN-NO             PIC S9(4) COMP VALUE +0.
     05  PARTNO-ERROR-SUBCODE                   PIC S9(4) COMP VALUE +0.

 01  PARTNAME-COMMAND-AREA.
      05  PARTNAME-COMMAND                      PIC XX VALUE SPACES.
     05  PARTNAME-TABLE                         PIC X(8) VALUE 'PARTNAME'.
     05  PARTNAME-FOUND                         PIC X VALUE SPACES.
     05  PARTNAME-INDIRECT-IND                  PIC X     VALUE LOW-VALUES.
     05  PARTNAME-RESERVED                      PIC X     VALUE LOW-VALUES.
     05  PARTNAME-ABEND-OVERRIDE                PIC X     VALUE SPACES.
     05  PARTNAME-ERROR-CODE                    PIC S9(4) COMP VALUE +0.
     05  PARTNAME-COUNT                         PIC S9(9) COMP VALUE +0.
     05  PARTNAME-LENGTHS.
         10  PARTNAME-ROW-OVERRIDE-LENGTH       PIC S9(9) COMP VALUE +0.
         10  PARTNAME-ROW-ACTUAL-LENGTH         PIC S9(9) COMP VALUE +0.
         10  PARTNAME-FG-KEY-LENGTH             PIC S9(4) COMP VALUE +0.
     05  PARTNAME-FUNCTION-ID                   PIC S9(4) COMP VALUE +0.
     05  PARTNAME-FUNCTION-AREA                 PIC X(8)  VALUE LOW-VALUES.
     05  PARTNAME-DATE-AREA           REDEFINES PARTNAME-FUNCTION-AREA.
         10  PARTNAME-DATE                      PIC X(8).
         10  RESERVED                           PIC X(20).
     05  PARTNAME-RETURNED-ABS-GEN-NO           PIC S9(4) COMP VALUE +0.
     05  PARTNAME-ERROR-SUBCODE                 PIC S9(4) COMP VALUE +0.

 01  PARTNAME-ALT-DEFINITION.
     05  PARTNAME-ORG                           PIC X VALUE 'S'.
     05  PARTNAME-METHOD                        PIC X VALUE 'B'.
     05  PARTNAME-KEY-COUNT                     PIC S9(4) COMP VALUE +1.
     05  PARTNAME-KEY-LOCATION                  PIC S9(9) COMP VALUE +11.
     05  PARTNAME-KEY-SIZE                      PIC S9(9) COMP VALUE +25.

 PROCEDURE DIVISION.
 HOUSEKEEPING.

***  GET PARTNO TABLE FROM TABLE BASE LIBRARY.

     MOVE 'OR' TO PARTNO-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          PARTNO-COMMAND-AREA

***  INVOKE THE ALTERNATE INDEX. THIS CAUSES THE ALTERNATE
***  INDEX TO BE CREATED ON THE PARTNO TABLE.

     MOVE 'IA' TO PARTNAME-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          PARTNAME-COMMAND-AREA
                          PARTNO-TABLE
                          PARTNAME-ALT-DEFINITION
```



```
 PROCESSING.

***  TO SEARCH FOR A VALID PARTNO WITH PART-NO AS A KEY, USE:

     MOVE 'SK' TO PARTNO-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          PARTNO-COMMAND-AREA
                          ORDER-PART-NUMBER

***  TO FETCH A PARTNO USING PART NAME AS A KEY INTO PARTNAME, USE:
     MOVE 'FK' TO PARTNAME-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          PARTNAME-COMMAND-AREA
                          PART-ROW
                          ORDER-PART-NAME
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX6C
 *
 *   Establish another key with which to search a table.
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szAltTableName[9] = "AARALT  ";
static char szStatus[6] = "NYYYN";
static char szReadPassword[2] = " ";
static int nGen = 0;
static char szKey[6] = "TIGER";
static char szSecondKey[16] = "123456789012345";
static int nRowSize = 20;
static int nKeySize = 5;
static int nSecondKeySize = 15;

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbCommandAreaStruct tbCommArea1;
   TbTableDefinitionStruct tbTableDef;
   TbAltDefinitionStruct  tbAltDef;
   char sReadPassword[8];
   char * pRowArea = NULL;
   char * pKeyArea = NULL;
   char * pSecondKeyArea = NULL;
   char sStatus[8];
```

```
char sTableName[8];
int nGeneration = nGen;

/*
 * Initialize the parameters.
 */
fixStringLength( szTableName, sTableName, 8 );
InitTbParm( &tbParm );
InitTbCommandArea( &tbCommArea, sTableName );
InitTbCommandArea( &tbCommArea1, sTableName );
InitTableDef( &tbTableDef );
InitAltDefinitionStruct( &tbAltDef );

/*
 * Initialize tableBASE with CS, ChangeStatus.
 */
fixStringLength( szStatus, sStatus, 8 );
memcpy( tbCommArea.tbCommand, "CS", 2 );
TBLBASE( &tbParm, &tbCommArea, sStatus );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "CS\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

tbAltDef.org = 'U';
tbAltDef.method = 'S';
tbAltDef.keyLocation = 6;
tbAltDef.keySize = 15;

/*
 * Open the table for read with OR.
 */
fixStringLength( szReadPassword, sReadPassword, 8 );
memcpy( tbCommArea.tbCommand, "OR", 2 );
TBLBASE( &tbParm, &tbCommArea, sReadPassword, nGeneration);
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "OW\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Create a temporary Alternate Index with IA (InvokeAlternate).
 * For performance reasons, it is better to use a different
 * TbCommandAreaStruct for each table and for each index.
 */
fixStringLength( szAltTableName, tbCommArea1.tbTable, 8 );
memcpy( tbCommArea1.tbCommand, "IA", 2 );
TBLBASE( &tbParm, &tbCommArea1, sTableName, &tbAltDef );
if( tbCommArea1.tbError != TB_SUCCESS )
```



```
{
   printf( "IA\n");
   printf( "Found code: %c\n", tbCommArea1.tbFound );
   printf( "Error code: %d\n", tbCommArea1.tbError );
   printf( "Sub code: %d\n", tbCommArea1.tbErrorSubcode );
   return tbCommArea1.tbError;
}

/*
 * Allocate spaces for the keys and the row.
 */
pRowArea = (char *) malloc( nRowSize + 1 );
if( pRowArea == NULL )
   return TB_ERROR;
memset( pRowArea, ' ', nRowSize );
pKeyArea = (char *) malloc( nKeySize );
if( pKeyArea == NULL )
{
   free( pRowArea );
   return TB_ERROR;
}
memset( pKeyArea, ' ', nKeySize );
pSecondKeyArea = (char *) malloc( nSecondKeySize );
if( pSecondKeyArea == NULL )
{
   free( pRowArea );
   free( pKeyArea );
   return TB_ERROR;
}
memset( pSecondKeyArea, ' ', nSecondKeySize );

/*
 * Search by Key (SK) using the Data Table.
 */
fixStringLength( szKey, pKeyArea, nKeySize );
memcpy( tbCommArea.tbCommand, "SK", 2 );
TBLBASE( &tbParm, &tbCommArea, pKeyArea );
if( tbCommArea.tbError != TB_SUCCESS || tbCommArea.tbFound != 'Y' )
{
   printf( "SK\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   free( pRowArea );
   free( pKeyArea );
   free( pSecondKeyArea );
   return tbCommArea.tbError;
}
else
{
   printf( "Found code: %c\n", tbCommArea.tbFound );
}

/*
 * Fetch by Key (FK) using the secondary index.
 */
fixStringLength( szSecondKey, pSecondKeyArea, nSecondKeySize );
```

```
   memcpy( tbCommArea1.tbCommand, "FK", 2 );
   /* Call TableBASE */
   TBLBASE( &tbParm, &tbCommArea1, pRowArea, pSecondKeyArea );
   if( tbCommArea1.tbError != TB_SUCCESS || tbCommArea1.tbFound != 'Y' )
   {
      printf( "FK\n");
      printf( "Found code: %c\n", tbCommArea1.tbFound );
      printf( "Error code: %d\n", tbCommArea1.tbError );
      printf( "Sub code: %d\n", tbCommArea1.tbErrorSubcode );
      free( pRowArea );
      free( pKeyArea );
      free( pSecondKeyArea );
      return tbCommArea1.tbError;
   }
   else
   {
      pRowArea[nSecondKeySize] = '\0';
      printf( "Row found by second key: %s\n", pRowArea );
   }

   if( pRowArea != NULL )
      free( pRowArea );
   if( pKeyArea != NULL )
      free( pKeyArea );
   if( pSecondKeyArea != NULL )
      free( pSecondKeyArea );

   return TB_SUCCESS;
}
```

Concatenate tableBASE libraries

The tableBASE test library is concatenated in front of the production library. If the test version of a table is present, it will be used.

In COBOL

```
 01  LIB-LIST.
     05  LIB-LIST-1              PIC X(8)  VALUE SPACES.
     05  LIB-LIST-2              PIC X(8)  VALUE SPACES.
     05  LIB-LIST-3-10           PIC X(64) VALUE SPACES.

 PROCEDURE DIVISION.
 HOUSE-KEEPING.
     MOVE 'TESTLIB' TO LIB-LIST-1
     MOVE 'PRODLIB' TO LIB-LIST-2
     MOVE 'ML'  TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA
                          LIB-LIST
```


In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX7C
 *
 *  Concatenate tableBASE libraries. The test library is
 *  put in front of the production library. If the test
 *  version of a table is present, it will be used.
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szStatus[6] = "NYYYN";
/*
 * Assume you have these DDName defined in your JCL.
 */
static char szTestLib[8] = "TESTLIB";
static char szProLib[7] = "PROLIB";

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbLibListStruct tbLibList;
   char sTestLib[8];
   char sProLib[8];
   char sTableName[8];
   char sStatus[8];

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );
   InitLibList( &tbLibList );

   /*
    * Initialize tableBASE with CS, ChangeStatus.
    */
   fixStringLength( szStatus, sStatus, 8 );
   memcpy( tbCommArea.tbCommand, "CS", 2 );
   TBLBASE( &tbParm, &tbCommArea, sStatus );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "CS\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }
```

```
   /*
    * Modify the library search order with ML.
    */
   fixStringLength( sTestLib, szTestLib, 8 );
   fixStringLength( sProLib, szProLib, 8 );
   memcpy( tbLibList.lib1, sTestLib, 8 );
   memcpy( tbLibList.lib2, sProLib, 8 );
   memcpy( tbCommArea.tbCommand, "ML", 2 );
   TBLBASE( &tbParm, &tbCommArea, &tbLibList );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "ML\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }

   return TB_SUCCESS;
}
```

Temporarily change a list of concatenated tableBASE libraries

It may occasionally be necessary to temporarily modify the concatenated library list and/or the status switches during processing, and their original values may not be known. Therefore, in order to restore them, the active library list and/or processing status must be saved first.

In COBOL

```
 DATA DIVISION.
 01  LIB-LIST-SAVE                  PIC X(80).
 01  STATUS-SWITCHES-SAVE           PIC X(8).

 01  LIB-LIST-NEW.
     05  LIBRARY-1                  PIC X(8)  VALUE 'TEST1'.
     05  LIBRARY-2                  PIC X(8)  VALUE 'TEST2'.
     05  LIBRARY-3                  PIC X(8)  VALUE 'TEST3'.
     05  LIBRARY-4                  PIC X(8)  VALUE 'PRODLIB'.
     05  FILLER                     PIC X(48) VALUE SPACES.

 01 STATUS-SWITCHES-NEW             PIC X(8)  VALUE 'YN'.


 PROCEDURE DIVISION.

***  SAVE CURRENT TABLE BASE LIBRARY LIST
     MOVE 'LL'  TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA
                          LIB-LIST-SAVE

***  SAVE CURRENT PROCESSING STATUS
```



```
     MOVE 'LS' TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA
                          STATUS-SWITCHES-SAVE

***  LOAD NEW TABLE BASE LIBRARY LIST
     MOVE 'ML'  TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA
                          LIB-LIST-NEW

***  LOAD NEW PROCESSING STATUS
     MOVE 'CS'  TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA
                          STATUS-SWITCHES-NEW
*    (normal processing until end of routine)

***  RESTORE PREVIOUS TABLE BASE LIBRARY LIST
     MOVE 'ML'  TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA
                          LIB-LIST-SAVE

***  RESTORE PREVIOUS PROCESSING STATUS
     MOVE 'CS'  TO xxxx-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          xxxx-COMMAND-AREA
                          STATUS-SWITCHES-SAVE
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX8C
 *
 *  Temporarily change a concatenated list of tableBASE libraries.
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szStatus[6] = "NYYYN";
/*
 * Assume you have these DDName defined in your JCL.
 */
```

```
static char szTestLib[8] = "TESTLIB";
static char szProLib[7] = "PROLIB";

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   TbLibListStruct tbLibListSaved;
   TbLibListStruct tbLibList;
   char sTestLib[8];
   char sProLib[8];
   char sTableName[8];
   char sStatus[8];
   char sStatusSaved[8];

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );
   InitLibList( &tbLibListSaved );
   InitLibList( &tbLibList );

   /*
    * Save the current library list with LL.
    */
   memcpy( tbCommArea.tbCommand, "LL", 2 );
   TBLBASE( &tbParm, &tbCommArea, &tbLibListSaved );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "LL\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }

   /*
    * Save the current status with LS.
    */
   memcpy( tbCommArea.tbCommand, "LS", 2 );
   TBLBASE( &tbParm, &tbCommArea, sStatusSaved );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "LS\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }

   /*
    * Change the library list with ML.
    */
   fixStringLength( sTestLib, szTestLib, 8 );
   fixStringLength( sProLib, szProLib, 8 );
```



```
memcpy( tbLibList.lib1, sTestLib, 8 );
memcpy( tbLibList.lib2, sProLib, 8 );
memcpy( tbCommArea.tbCommand, "ML", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbLibList );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "ML\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Change to the new status with CS.
 */
fixStringLength( szStatus, sStatus, 8 );
memcpy( tbCommArea.tbCommand, "CS", 2 );
TBLBASE( &tbParm, &tbCommArea, sStatus );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "CS\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

/*
 * Insert all the normal processing here.
 */

/*
 * At the end of normal processing restore both status
 * and library search order.
 */

memcpy( tbCommArea.tbCommand, "ML", 2 );
TBLBASE( &tbParm, &tbCommArea, &tbLibListSaved );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "ML\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
}

memcpy( tbCommArea.tbCommand, "CS", 2 );
TBLBASE( &tbParm, &tbCommArea, sStatusSaved );
if( tbCommArea.tbError != TB_SUCCESS )
{
   printf( "CS\n");
   printf( "Found code: %c\n", tbCommArea.tbFound );
   printf( "Error code: %d\n", tbCommArea.tbError );
   printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
   return tbCommArea.tbError;
```



```
    }

    printf( "Restored the original LibList and Status-switch\n");

    return TB_SUCCESS;
}
```

Access a table indirectly

The application will access the appropriate rate table indirectly. This feature of tableBASE is called Open Indirect (see "Open Indirect" on page 61).

Open Indirect causes a table to be searched for a target secondary table name. This secondary table is the one that will be opened if the search of the primary table is successful. The primary or first table may not be read password protected. The first table has only two fields:

TARGET-TABLE-NAME(8 bytes)

INDIRECT-OPEN-CRITERION(up to 50 bytes)

The second field is the key of the first table. For example, each key field might represent a date when the target rate table in TARGET-TABLE-NAME is to go into effect. The Set Indirect (SI) command is used to provide the processing date that becomes the search argument for the primary table. A search of the primary table is successful if the search argument is equal to a row-key, or is greater than one row-key, and less than the next row-key, of the primary table. After a successful Indirect Open operation, the target table name is placed in the TABLE field of the COMMAND-AREA, and the error code is 0. The FOUND code is N, whether the search was successful or not.

Abend processing will be suppressed in this example, so all calls to tableBASE must be followed by a test of the error code.

In COBOL

```
 DATA DIVISION.
 01  INPUT-RECORD.
   05  .....
   05  INPUT-PROCESSING-DATE           PIC X(8).
   05  INPUT-CODE .....

 01  STATUS-SWITCHES.
   05  ST-ABEND                        PIC X.
   05  FILLER                          PIC X(7)  VALUE SPACES.

 01  RATE-ROW.
   05  RATE-CODE.....
   05  RATE-VALUE....

 01  RATE-COMMAND-AREA.
   05  RATE-COMMAND                    PIC XX    VALUE SPACES.
   05  RATE-TABLE                      PIC X(8)  VALUE SPACES.
```



```
   05  RATE-FOUND                       PIC X     VALUE SPACES.
   05  RATE-INDIRECT-IND                PIC X     VALUE LOW-VALUES.
   05  RATE-RESERVED                    PIC X     VALUE LOW-VALUES.
   05  RATE-ABEND-OVERRIDE              PIC X     VALUE SPACES.
   05  RATE-ERROR-CODE                  PIC S9(4) COMP VALUE +0.
   05  RATE-COUNT                       PIC S9(9) COMP VALUE +0.
   05  RATE-LOCK                        PIC X(8)  VALUE SPACES.
   05  RATE-LENGTHS.
       10  RATE-ROW-OVERRIDE-LENGTH     PIC S9(9) COMP VALUE +0.
       10  RATE-ROW-ACTUAL-LENGTH       PIC S9(9) COMP VALUE +0.
       10  RATE-FG-KEY-LENGTH           PIC S9(4) COMP VALUE +0.
   05  RATE-FUNCTION-ID                 PIC S9(4) COMP VALUE +0.
   05  RATE-FUNCTION-AREA.
       10  RATE-DATE                    PIC X(8).
       10  FILLER                       PIC X(20).
   05  RATE-RETURNED-ABS-GEN-NO         PIC S9(4) COMP VALUE +0.
   05  RATE-ERROR-SUBCODE               PIC S9(4) COMP VALUE +0.

***
*    THE FOLLOWING IS A COPY OF CONTENTS OF PRIMARY RATETBL.
*    IT IS PRESENTED HERE ONLY TO COMPLETE THE UNDERSTANDING
*    OF TABLEBASE WHEN PROCESSING THE PRIMARY RATETBL.
***

 01  RATETBL-PRIMARY-ROW.
   05  RATETBL-SECONDARY                PIC X(8) VALUE 'RATE2000'.
   05  RATETBL-KEY                      PIC X(8) VALUE '20000101'.

 PROCEDURE DIVISION.
 HOUSE-KEEPING.

***  TURN  OFF  TABLEBASE  ABEND  FUNCTION.
     MOVE 'N' TO ST-ABEND
     MOVE 'CS' TO RATE-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                  RATE-COMMAND-AREA
                  STATUS-SWITCHES

***  GET TODAY'S DATE AND MOVE IT TO INPUT-PROCESSING-DATE
***  SET  UP  VALUE  TO  BE  USED  FOR  INDIRECT  TABLE  OPEN

     MOVE 'SI' TO RATE-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                  RATE-COMMAND-AREA
                  INPUT-PROCESSING-DATE
     IF RATE-ERROR-CODE GREATER THAN ZERO
       GO TO TABLEBASE-ERROR-RTN
     ENF-IF
***  OPEN INDIRECT TABLE.

     MOVE 'I'  TO RATE-INDIRECT-IND
     MOVE 'RATETBL'  TO RATE-TABLE.
     MOVE 'OR' TO RATE-COMMAND
***
*    THE PRIMARY RATE IS SEARCHED USING TODAY'S DATE AS A KEY.
*    SINCE IT IS GREATER THAN 20000101, THE ONLY ENTRY IN THE
*    PRIMARY TABLE IS FOUND AND THE NAME 'RATE2000' IS PLACED
```

```
*    IN THE RATE-COMMAND-AREA BY TABLEBASE AND IS OPENED READY FOR
*    FURTHER PROCESSING.
***
     CALL 'TBLBASE' USING TB-PARM
                          RATE-COMMAND-AREA
     IF RATE-ERROR-CODE GREATER THAN ZERO
        GO TO TABLEBASE-ERROR-RTN
     ENDIF
 GET-RECORD.

*      (get input record)

 PROCESS-ROW.
     MOVE 'FK' TO RATE-COMMAND
     CALL 'TBLBASE' USING TB-PARM
                          RATE-COMMAND-AREA
                          RATE-ROW
                          INPUT-CODE
     IF RATE-ERROR-CODE GREATER THAN ZERO
        GO TO TABLEBASE-ERROR-RTN
*      (correct rate table for processing date will be used)
     END-IF
```

In C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "dkh.h"

/*
 *  DK1TEX9C
 *
 *  Access a Table Indirectly.
 *
 *  The primary table contains the names of the secondary
 *  tables (to be open indirectly) and a key used to make
 *  the selection. A row of this primary table might be
 *  expressed with this C struct:
 *
 *  typedef struct
 *  {
 *     char tblname_secondary [8];
 *     char selection_criteria [20];
 *  } PrimaryRow;
 *
 *  The precise table that will be open indirectly is the
 *  one with the largest kyt equal to or less than the value
 *  specified in the selection_criteria field.
 */

/*
 * Assume these are user inputs.
 */
static char szTableName[6] = "AARON";
static char szStatus[6] = "NYYYN";
static char szIndirectOpenCriterion[11] = "2003-04-21";
```



```
static char szReadPassword[2] = " ";
static int nGen = 0;
static int nRowSize = 20;
static int nKeySize = 5;
static int nKeyLocation = 1;

int main(void)
{
   TbParmStruct tbParm;
   TbCommandAreaStruct tbCommArea;
   char sReadPassword[8];
   char sStatus[8];
   char sTableName[8];
   char sIndirectOpenCriterion[50];
   int nGeneration = nGen;
   char * pRowArea = NULL;

   /*
    * Initialize the parameters.
    */
   fixStringLength( szTableName, sTableName, 8 );
   InitTbParm( &tbParm );
   InitTbCommandArea( &tbCommArea, sTableName );

   /*
    * Initialize tableBASE with CS, ChangeStatus.
    */
   fixStringLength( szStatus, sStatus, 8 );
   memcpy( tbCommArea.tbCommand, "CS", 2 );
   TBLBASE( &tbParm, &tbCommArea, sStatus );
   if( tbCommArea.tbError != TB_SUCCESS )
   {
      printf( "CS\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }

   /*
    * Setup the indirect open with SI.
    */
   fixStringLength( szIndirectOpenCriterion,
      sIndirectOpenCriterion, 50 );
   memcpy( tbCommArea.tbCommand, "SI", 2 );
   TBLBASE( &tbParm, &tbCommArea, sIndirectOpenCriterion );
   if( tbCommArea.tbError == TB_SUCCESS )
   {
      /*
       * Open the indirect table (the tbCommArea.tbTable field
       * was set to the primary table previously).
       * If the key (selection criteria) is found (or the greatest key
       * less than the criteria is found), the corresponding table
       * will be open and the field tbCommArea.tbTable will be
       * updated with the actual name of the open table.
       */
      fixStringLength( szReadPassword, sReadPassword, 8 );
```

```
      memcpy( tbCommArea.tbCommand, "OR", 2 );
      tbCommArea.tbIndirectOpen = 'I';
      TBLBASE( &tbParm, &tbCommArea, sReadPassword, nGeneration);
      if( tbCommArea.tbError == TB_SUCCESS )
      {
         /*
          * Get the first row (GF).
          */
         pRowArea = (char *) malloc( nRowSize + 1 );
         if( pRowArea == NULL )
            return TB_ERROR;
         memset( pRowArea, ' ', nRowSize );

         fixStringLength( szReadPassword, sReadPassword, 8 );
         memcpy( tbCommArea.tbCommand, "GF", 2 );
         TBLBASE( &tbParm, &tbCommArea, pRowArea );
         if( tbCommArea.tbError != TB_SUCCESS )
         {
            printf( "GF\n");
            printf( "Found code: %c\n", tbCommArea.tbFound );
            printf( "Error code: %d\n", tbCommArea.tbError );
            printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
            free( pRowArea );
            return tbCommArea.tbError;
         }

         /*
          * Print the output.
          */
         pRowArea[nRowSize] = '\0';
         printf( "Row: %s\n", pRowArea );

         /*
          * Close the table (CL).
          */
         memcpy( tbCommArea.tbCommand, "CL", 2 );
         TBLBASE( &tbParm, &tbCommArea );
         if( tbCommArea.tbError != TB_SUCCESS )
         {
            printf( "CL\n");
            printf( "Found code: %c\n", tbCommArea.tbFound );
            printf( "Error code: %d\n", tbCommArea.tbError );
            printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
            return tbCommArea.tbError;
         }
      }

   }
   else
   {
      printf( "SI\n");
      printf( "Found code: %c\n", tbCommArea.tbFound );
      printf( "Error code: %d\n", tbCommArea.tbError );
      printf( "Sub code: %d\n", tbCommArea.tbErrorSubcode );
      return tbCommArea.tbError;
   }
```



```
   if( pRowArea != NULL )
      free( pRowArea );

   return TB_SUCCESS;
}
```

Other language examples

The following example illustrates how to define TBLBASE parameters and access a table in ASSEMBLER language.

ASSEMBLER coding example

```
*    TBparm area for tblBase
TBParm        DS    0cl64    <<--- Release 5 and after TB-Parm
TBp_id        dc    cl2'TB'        TB-Parm id
              dc    xl2'00'        Must be set to binary zero
TBp_version   dc    c'5'           TB-Parm format is Rel 5 and after
TBp_format    dc    c'0'           '0' means use 72-byte Command Areas
              dc    xl18'00'       Reserved (should be binary zero)
TBp_subsystem dc    xl4'00'        VTS name if used else binary zeroes
              dc    xl8'00'        Reserved (should be binary zeroes)
TBp_turbo     dc    xl8'00'        Turbo (set to zero for 1st call)
              dc    xl16           Reserved (should be binary zeroes)

*    Command area for tblBase
Command_Area   DS    0cl72   <<--- Long-form Command area
Cmd_command    ds    cl2            Command to be performed by tableBASE
Cmd_table      ds    cl8            Name of table to be used
Cmd_found      ds    c              Found code
Cmd_indirect   dc    c              Indirect Open indicator
               dc    x'00'          Reserved - do not alter
Cmd_abend_override   dc    c' '     Abend override
Cmd_error_code ds    h              Error code set by tblBase
Cmd_count      dc    f'0'           Count field set by tblBase
Cmd_lock_latch dc    xl8'0'         Lock-latch
Cmd_row_length_override dc   f'0'  Row length override set by caller
Cmd_row_actual_length   ds   f     Actual row length returned
Cmd_FG_key_length       dc   h'0'  Fetch Generic partial key length
Cmd_function_id         dc   h'0'  Special processing value if not zero
Cmd_dsp_date   dc    xl8'0'         Date for Date-Sensitive Processing
               dc    cl20           Reserved (should be binary zero)
Cmd_rtn_abs_gen ds   h              Returned Absolute Generation set by tblBase
Cmd_subcode    ds    h              Error subcode set by tblBase

*    Extended DT block for GD and DT commands
Definition_block    DS    0cl256    <<--- Long-form Definition Block
Dt_Organization     dc    cl1'S'
Dt_Search_method    dc    cl1'B'
Dt_Index            dc    cl1'P'
Dt_SMC              dc    cl1'R'
```



```
Dt_Read_Password    dc    cl8' '
Dt_Write_Password   dc    cl8'SECURITY'
Dt_Row_size         dc    f'120'
Dt_Key_size         dc    f'11'
Dt_Key_location     dc    f'1'
Dt_Number_of_rows   dc    f'500'
Dt_Generations      dc    h'3'
Dt_Expansion_factor dc    h'0'
Dt_Low_density      dc    h'0'
Dt_High_density     dc    h'0'
                    ds    xl6
Dt_Date_time        ds    cl12
Dt_Absolute_generation_number  dc    h'0'
Dt_Dataset_name     ds    cl44
Dt_Relative_generation_number  dc    h'0'
Dt_Generations_present   dc    h'0'
Dt_Rows_at_expansion     dc    f'0'
Dt_DDname           ds    cl8
Dt_Data_table       dc    cl8' '
Dt_Open_status      ds    c
Dt_Alternates_invoked    dc    c' '
Dt_View_version     dc    cl1'5'
                    ds    x
Dt_Userid           ds    cl8
Dt_View_name        ds    cl8
Dt_View_date        ds    cl12
Dt_User_comments    dc    cl16' '
                    ds    cl76

*   Library list
Liblist     DS    (0*10)cl8
            dc    cl8'PROGLIB'
            dc    cl8'SYSTMLIB'
            dc    cl8'INSTALIB'
            dc    7*cl8' '


*    Row area with key
Row_Area   ds    0cl120
Row_key    ds    cl11
Row_data   ds    cl109


*   Some test keys
testkeys ds   0cl11
         dc   cl11'aaaaaaa'
         dc   cl11'bbbbbbb'
numkeys  equ  (*-testkeys)/l'testkeys
         ...
         ...
         ...
*   Set up the lbrary concatenation order.
         mvc   Cmd_command,=c'ML'
         Call  tblBase,(TBparm,Command_Area,Liblist),VL
         clc   Cmd_error,=h'0'
         bne   MLerror
```

```
*   Define the new table.
         mvc   Cmd_command,=c'DT'
         mvc   Cmd_table,=cl8'TABLE01'
         Call  tblBase,(TBparm,Command_Area,DT_definition_block),VL
         clc   Cmd_error,=h'0'
         bne   DTerror

*   Set up command outside loop.
         mvc   Cmd_commmand,=c'IK'

C   Read row records until end-of-file, and insert each into the table.
getrow   get   infile,Row

*   Search and insert into table
         Call  tblBase,(TBparm,Command_Area,Row_Area),VL
         clc   Cmd_error,=h'0'
         bne   IKerror
         cli   Cmd_found,c'Y'
         bne   getrow

*   Log duplicate key.
         mvc   dup_msg_key, row_key
         put   logfile,dup_msg
         b     getrow

*   Store new table on PROGLIB, the first entry in the ML list.
RowEOD   mvc   Cmd_command,=c'ST'
         Call  tblBase,(TBparm,Command_Area),VL
         clc   Cmd_error,=h'0'
         bne   STerror

*   Look up some sample keys.
         la    r2,testkeys
         la    r3,numkeys
         mvc   Cmd_command,=c'SK'

samploop Call  tblBase,(TBparm,Comand_area,(r2)),VL
         clc   Cmd_error,=h'0'
         bne   SKerror

         mvc   msg_word,=cl9'Found'
         cli   Cmd_found,c'Y'
         be    *+10
         mvc   msg_word,=cl9'Not found'

         put   logfile,samp_msg
         la    r2,l'testkey(,r2)
         bct   r3,samploop

C   Close the table.
         mvc   Cmd_Command,=c'CL'
         Call  tblBase,(TBparm,Command_Area),VL
         clc   Cmd_Error,=h'0'
         bne   Closerr
         ...
         ...
*  Routines to handle errors in various commands
```

```
MLerror  ds   0h
DTerror  ds   0h
IKerror  ds   0h
SToserr  ds   0h
SKerror  ds   0h
CLoserr  ds   0h
         ...
         ...
         ...
infile   dcb  ddname=INFILE,dsorg=PS,macrf=GM,recfm=FB,lrecl=120,      X
              eodad=RowEOD
logfile  dcb  .....

dup_msg  dc   c' Duplicate key '
dup_key  ds   cl11

samp_msg dc   c' Test key '
samp_key ds   cl11
         dc   c' '
msg_word ds  cl9
```

6 Environmental interfaces

The documentation in this guide focuses on the TBLBASE API to tableBASE. This interface operates in batch, CICS, and IMS TM environments, with tables in either a local TSR or shared TSR (VTS-TSR).

When a tableBASE table is opened in any environment, storage is allocated within a TSR and a copy of the table is loaded into the TSR from the tableBASE library. The TSR is either a local Data Space allocated at region initialization or a shared Data Space allocated during VTS Agent initialization. Only one region may have a table open for write but a table may be opened for read in multiple regions.

The following sections describe the operation of TBLBASE in each supported environment.

Batch environment

MVS compile and link JCL requirements

It is recommended that the TBLBASE stub be statically linked with the application program at Link-edit or Bind Time.

Link JCL should include the following:

```
//SYSLIB    DD DSN=your.dsn.qualifier.LOAD,DISP=SHR
```

MVS run-time JCL requirements

If the tableBASE system load library is not included in the MVS linklist, then runtime JCL should include the following:

```
//STEPLIB    DD DSN=your.dsn.qualifier.LOAD,DISP=SHR
```


tableBASE libraries may be allocated to a batch region with a disposition of OLD if exclusive control of the library or tables is required, or SHR if tables can be shared with other regions (such as CICS).

To reference an existing tableBASE library, use this JCL as a pattern:

```
//DDNAME    DD DSN=your.prefix.TBASE.MAINLIB,DISP=SHR
```

Each tableBASE library is identified by a unique DDNAME. You may not associate more than one DDNAME with the same tableBASE library. Multiple libraries cannot be concatenated using a single DD statement. The tableBASE ML command is available for logically concatenating libraries.

tableBASE defaults to a library search list consisting of the single DDNAME MAINLIB, unless a different default is specified in the TBOPT file (see "Parameters for all tableBASE environments" on page 242). The ML command may be used to change the tableBASE library search list to any set of libraries required.

If a TBOPT file will be used as a source of run-time parameter input, a TBOPT DD statement must be provided. For more information about using a TBOPT dataset, see "TBOPT dataset coding" on page 405.

If a tableSPACE report (STROBE report) is required, a DD statement must be provided with a DDNAME of TBTSRPT.

Note: The record length for TBTSRPT is 80 characters.

TBLBASE

Although we recommend static linking, TBLBASE may be linked either statically or dynamically. In either case, TBLBASE itself dynamically loads other tableBASE modules as appropriate. For this reason, the load library containing tableBASE must be available to the calling program, either through the system linklist or through the JOBLIB or STEPLIB JCL statements.

The batch interface may also be used in TSO. In this case, if the calling program is running in foreground, the tableBASE load library must be identified in the system link list or it must be allocated in the User's LOGON procedure.

TBLBASE is re-entrant, which means that TBLBASE code and tables are shared by all modules calling within a given job step. Although the TBLBASE stub, as shipped, is link-edited as Amode (ANY) Rmode (24)—which was done for backwards compatibility—the stub can be safely link-edited into an Amode(31), Rmode(any) application module.

Termination processing

When processing is finished, there are certain end-of-processing activities which must be performed by tableBASE.

Under MVS batch, when tableBASE detects a normal return to the operating system signifying end-of-processing, it will automatically ensure that end-of-processing activities are executed. These activities include:

- closing tableBASE libraries that are opened
- optionally producing a tableSPACE Report showing the statistics accumulated since the last strobe, or since the job was initiated if a full strobe interval has not elapsed.

If the system abends, tableBASE may not be able to perform normal termination processing.

Online and multitasking environment considerations

When accessing tables in an online transaction-oriented environment, or multitasking environment such as a DB2 stored procedure address space (SPAS), the commands ML and CS operate on a transaction basis. Each transaction is initialized with a default library search order (LIB-LIST), default status switches, and VTSNAME, but has the ability to set and modify these settings. Each transaction is considered as a distinct thread of execution; the settings used in one transaction are not retained for the next transaction.

Sharing tables

The following scenarios are presented to illustrate sharing tableBASE tables between transactions in a single CICS or IMS TM region or a multitasking batch region:

- Retrieval only
- Temporary updating
- True updating
- Posting.

Retrieval only

Multiple terminals, transactions and/or applications wish to share access to the same table on a retrieval only basis.

Each user may issue an Open Read (OR) command for the table, or take advantage of the auto-open facility. tableBASE will provide a single, common copy of the table for all users to share in the local TSR.

Note: For tables opened with an Open Read (OR) command in a shared environment, updating commands are not allowed. An error code (0073) will be returned on any attempt to modify the memory resident image of a table which is opened for read.

Note: After a CN, the original name is no longer open. It may be opened later with an OR or OW, or an implicit open.

Temporary updating

A user wants a private copy of a table to update temporarily, without storing it. The user must change the name of the table (CN command) to some unique name, perhaps based upon the UserID or Terminal ID.

If the Change Name is successful, the user has his own exclusive copy of the table. For more details see "Change Name (CN)" on page 70.

Note: tableBASE will not allow this for a table which is already open for write by another user with a different LOCK-LATCH password, and will issue an error code (0074). Should this error occur, the user must decide whether to abort or to wait and try again.

True updating: sharing between retrieval and update

A user wishes to access a table in order to modify its contents and store them, a single intent that may or may not consist of several transactions. For example, the normal flow of transactions might be an open for write, multiple update transactions, and a final transaction to cause the table to be stored (ST) and closed (CL).

Obtain an exclusive copy

To obtain an exclusive copy of a table which may be retained for several transactions, the user must issue an Open Write command. If the table is already opened for read, tableBASE will verify that the generation on the library is the same as that in memory. If it is the same generation, tableBASE will immediately open the table for write.

Any subsequent attempts to open a table for write when it is already opened for write will be returned to the application with a non-zero ERROR code (or an abend if that option is selected). Once the original transaction (or user with multiple transactions) has completed updating, the table can be released with the RL or CL command so that another user/transaction in another region can proceed with updating.

Protecting an open table

While a password can be used to provide security to a table in a library, the LOCK-LATCH password provides limited security for an open table. See page 146 for more information.

Some examples of the use of the LOCK-LATCH field follow:

- If it is desired to share a table for updating, such as a posting table described in the Posting scenario below, the LOCK-LATCH password can be set to a value known by all users in the group.
- If it is desired that a table be updated by only one application at a time, although by as many tasks and transactions within that application as desired, the LOCK-LATCH password can be set to some application specific identifier.
- If it is desired that only tasks which are the same transactions should be updating the table simultaneously, the LOCK-LATCH password can be the Transaction ID.
- Alternatively, the LOCK-LATCH password could be set to a particular USERid to ensure exclusive control.

Example: posting

Multiple users wish to update a table at the same time. In this scenario, the open for write can be performed once by a transaction at the beginning of the day, and postings can be made throughout the day with periodic storing (ST command). The table can be opened either with or without a LOCK-LATCH password that allows multiple users to update the table.

Since the table may be shared among multiple update transactions, to prevent interference there must be application specific enqueues against the table while an update is going on. These must be performed by each user in each user program.

Warning: If a table is being updated by more than one user at a time, the IC, RC, and DC commands should not be used. These commands rely on a previously determined position (usually set by a Fetch by Key request) that may be changed by another user inserting or deleting a row in the same table.

CICS interface

To tableBASE, CICS is a single region or task. Only one copy of a table can be open in a region at any time. This copy of the table will be shared by all users (transactions). Read the section "Online and multitasking environment considerations" on page 231 before using tableBASE in a CICS environment.

The user should understand the following:

- tableBASE manages tables in-memory, and consequently, relies on the user to store (ST) any updated tables.
- If for any reason the CICS region abends or comes down, any table updates not yet stored are lost.

Restrictions

TBEXEC cannot be run under CICS.

No tableBASE library initialization (DL command) is possible under CICS. It must be performed by means of a batch run of TBEXEC.

TBLBASE will produce the tableSPACE Report data on the Journal File when a CICSJRNL is 99 (assuming STROBE= is not set to zero). This may be changed by the tableBASE administrator.

Some commands that work in a batch environment are not supported in the CICS online environment. These are:

- AL – Allocate Library
- UL – De-allocate Library
- DL – Define library.

Default STATUS SWITCHES

For the CICS interface to tableBASE, the default values for the STATUS SWITCHES include ABEND=N and WAIT=N. These and other settings can be altered at product installation time or later by the tableBASE administrator or by parameter settings via a TBOPT file.

LOCK-LATCH

The LOCK-LATCH field in the command area must be set to a unique value when a table is opened for write if the user wishes to maintain exclusive update control of the table over successive transactions. See page 146 for more information.

If the LOCK-LATCH password is lost or forgotten, the table can be accessed with the master password in the LOCK-LATCH field. See your tableBASE administrator for more information.

TBLBASE

It is recommended that TBLBASE be statically linked, and not defined to CICS. By statically linking TBLBASE, the PPT is not searched for TBLBASE and the call is very efficient. If it is called dynamically, it must be defined to CICS.

All other tableBASE modules are loaded and kept resident under CICS; a single copy serves all users.

TBLBASE is supported under the CEDF transaction allowing tableBASE calls to be traced and debugged.

CICS JCL requirements

All tableBASE libraries required by the CICS application must be placed in the CICS File Control Table (FCT) or the CICS System Definition (CSD).

Each tableBASE library is identified by a unique DDNAME. You may not associate more than one DDNAME with the same tableBASE library. Multiple libraries cannot be concatenated using a single DD statement. The tableBASE ML command is available for logically concatenating libraries.

The remainder of the JCL for running tableBASE under CICS is usually the responsibility of the installation's CICS Support Group, and is described in the installation instructions provided when the product is delivered. For more information, see the *tableBASE Installation Guide*.

IMS TM interface

To tableBASE, each IMS MPR is a single region or task. Therefore, with an MPR, only one copy of a table can be open at any time. This copy of the table will be shared by all transactions that run in a particular MPR.

The user should understand the following:

- tableBASE manages tables in-memory, and consequently, relies on the user to store (ST) any updated tables.
- If for any reason the MPR abends or comes down, any table updates not yet stored are lost.
- If any transaction abends in an MPR, all pre-loaded programs are reloaded. The TSR is deleted and recreated and the tables will be reloaded if they are needed. From a

performance point of view, this slows down the throughput of the MPR. From a tableBASE update point of view, any table updates not yet stored are lost.

- While a table is open for update in an MPR, it cannot be opened for update in another MPR. Therefore, the user must ensure that all tableBASE update transactions for a particular table be run in the same MPR.

One possible way for an application to detect the loss of table updates would be to establish three transactions: one to open the table being updated, one to update it, and one to store it. At the start of the day or at the beginning of an updating session, a transaction should be written to do an open write against the table, expecting an error to be returned because the table is already open. If the return code is zero, the table was not open. This indicates that the region had been down and some sort of recovery may be needed.

Restrictions

Some commands are not supported in the IMS TM environment. These commands are:

- AL – Allocate library
- UL – De-allocate library
- DL – Define library.

Default STATUS SWITCHES

For the IMS TM interface to tableBASE, the default value of the STATUS SWITCHES are ABEND=N and WAIT=N. These delivered defaults can be altered at product installation time or at run time via a TBOPT file.

TBLBASE

TBLBASE, the tableBASE API, determines what type of environment it is being executed in, and if it is not an IMS MPR, it performs exactly the same as it would in batch. Message processing programs (MPPs) to be tested using IBM's Batch Terminal Simulator (BTS) can be run in the IMS MPRs without having to relink the MPPs.

TBLBASE should be statically link-edited with each MPP wishing to call tableBASE.

Each time an MPP is loaded into an MPR, TBLBASE considers it as a distinct thread of execution and initializes it with a fresh copy of the default LIB-LIST, status switches, and VTSNAME. This means that if an MPP is used for more than one transaction before it is reloaded, it must manage the status switches and library search order, LIB-LIST, between transactions.

DB2 stored procedures

DB2 Stored Procedures managed either by DB2 or by WorkLoad Manager (WLM) are supported by tableBASE Version 6. The Stored Procedures Address Space (SPAS) environment is similar to using tableBASE in a multitasking batch environment (see "Online and multitasking environment considerations" on page 231).

Tables that are in a local TSR are not shared between Stored Procedures Address Spaces. A VTS-TSR should be used for most stored procedures.

DataKinetics recommends that the local TSR not be used for tables created or accessed by stored procedures, since WLM determines the address space(s) in which a stored procedure executes.

The user should understand the following:

- tableBASE manages tables in-memory, and consequently, relies on the user to store (ST) any updated tables.
- If for any reason the VTS-TSR being used by the SPAS abends or comes down, any table updates not yet stored are lost. However, if the SPAS ends normally or abnormally, the table updates are still available in the VTS-TSR and may be stored by transactions initiated after the SPAS is re-started.

All DB2 Stored Procedures accessing tableBASE must be linked AMODE(31), RMODE(ANY).

Run-time JCL for SPAS is similar to the JCL stream used in tableBASE batch environments (see "MVS run-time JCL requirements" on page 229), and should include DDs for any tableBASE libraries to be accessed by applications (DB2 Stored Procedures).

Note: The first DB2 Stored Procedure to call tableBASE initializes tableBASE in SPAS. If for some reason this initial stored procedure abends, all other stored procedures that are currently accessing tableBASE may be at risk of abend and the local TSR and its contents will be lost. This risk is very low. If an abend were to occur, the next stored procedure to call tableBASE will cause tableBASE to load itself and set up a new TSR.

Restrictions

Strobe is not supported for use with DB2 stored procedures; set Strobe = 0.

Some commands are not supported in the SPAS environment. These commands are:

- AL – Allocate library
- UL – De-allocate library
- DL – Define library.

Virtual Table Share (VTS-TSR)

A VTS-TSR allows multiple applications, co-existing within the same operating system (multiple different MVS jobs), to access tables that reside in a shared TSR. This allows applications to share tables. With Version 6, these tables can be opened for read or write. Access to tableBASE VTS is supported under MVS batch, TSO, IMS, CICS and DB2.

The VTS interface has two components:

- The shared TSR owner job, the VTS Agent
- The user applications access interface, the VTS user.

Note: With previous releases of tableBASE, the use of the TBCALLV was promoted as way to access VTS with the shortest path. This is no longer the case with Version 6. The use of TBCALLV offers no advantages in Version 6.

VTS management

VTS-TRSs are managed by a tableBASE process. If your installation includes both tableBASE VTS and tableBASE Process Manager, your VTS-TSRs are managed by VTS Managers (TPVMs); if your installation includes only tableBASE VTS, they are managed by the VTS Agent. In order to create a tableBASE Virtual TSR (VTS-TSR) a VTS Agent must be started, or tableBASE Process Manager must be running.

The start-up job to create a VTS-TSR may be included in the installation's IPL-time start-up procedures but it must start after either the Program Call Server job/STC, or the tableBASE Process Manager has been started. For more information on the Program Call Server/VTS Agent and the tableBASE Process Manager, see the *tableBASE Administration Guide*.

Note: The VTS Agent was referred to as the VTS Server in previous releases.

Each VTS Agent or VTS Manager is identified by a unique eight-character VTS-TSR name defined to MVS. This name is used to provide a logical connection between the VTS-TSR and the VTS user. Each VTS-TSR may be defined to hold a maximum of 2G. Tables can be loaded with a utility (TBDRIVER) or by any region with access to the VTS-TSR.

A VTS user application may refer to any number of VTS-TSRs concurrently.

Should the VTS Agent or VTS Manager abend during execution for any reason, the Operations support staff may re-submit the start-up job to MVS for re-initialization. During this time, the VTS user applications are not able to access the VTS-TSR, and an error code is returned indicating that the VTS-TSR is unavailable at this time.

Note: A complete understanding of the function of the VTS Agent or VTS Manager is not a requirement for the application developer for whom this manual is

intended. For more complete information, see the *tableBASE Administration Guide*.

The VTS User

The VTS User interface is incorporated into TBLBASE (the common tableBASE interface) and access to the VTS-TSR is transparent for all environments. The user applications may access the VTS-TSR using standard TBLBASE calls as described in Chapter 3 “tableBASE commands” on page 57.

Connecting a VTS User to a VTS-TSR

There are two methods to identify the VTS-TSRs that you which to access:

- Specify the VTS-TSR name in the TB-SUBSYSTEM field of TB-PARM. This allows you to access that VTS-TSR directly without going through the local TSR. If this field is blank or low-values, the access defaults to the local TSR. This method allows read-write access to the VTS-TSR. This is referred to as direct access.
- Specify the VTS-TSR names in the LIB-LIST. Instead of using a DDNAME, use VTS:xxxx where xxxx is the VTS-TSR name. In CICS, the use of the LIB-LIST applies only for the duration of the transaction for which the ML command is issued. When a table is first opened, the LIB-LIST is searched. If a VTS-TSR name is encountered, the VTS-TSR is searched and if the table is found, a link is established and the table is flagged as being open-for-read in the local TSR. This method allows read-only access to the VTS-TSR (as in releases prior to Version 6 of tableBASE). This is referred to as a linked access.

There are several parameters (VTSFIRST, VTSLAST, VTSNAME) that can alter the way a VTS-TSR is accessed; these parameters have default values set at the time tableBASE is installed but run-time settings for them may be provided by a TBOPT file. See Chapter 7 “Customizing tableBASE options” on page 241 to identify VTS-TSRs to be added to the LIB-LIST. The VTSFIRST=xxxx parameter identifies a VTS-TSR to be searched before the first library (or VTS-TSR) in the LIB-LIST; similarly a VTSLAST=xxxx parameter is available to add another VTS-TSR to the search list after the last library (or VTS-TSR) specified in the LIB-LIST. This method allows read-only access to the VTS-TSR (as in releases prior to Version 6 of tableBASE).

VTS User access

In order to access the data in the VTS-TSR, the application program contains typical tableBASE calls. It is through parameters or library lists that tableBASE knows VTS should be invoked. The TBLBASE API supports access to tableBASE VTS-TSR under MVS batch, TSO, IMS, CICS and DB2.

The tableBASE options for applications using TBCALLV or TBASEV may have changed. In Version 5 the options were obtained from the VTS setting in CV113450 (linked into

CVROOTV). In Version 6 the options are obtained from the region in which the application runs (DK1T1134 for batch, DK1T2734 for CICS).

Specifically, in Version 5 the default SWITCHES option was "NNNNN" for VTS while it was "YNYYN" for batch. The change in the switch settings can mean behavioural changes in applications using TBCALLV/TBASEV. For example, an application which processed a non-zero tableBASE error code would now abend (ABEND SWITCH=Y). This difference can normally be resolved by adding a TBOPT override to the application JCL (see Appendix C, "tableBASE run-time options" on page 389).

VTS User messages

All tableBASE/VTS user error messages are the same as the normal errors issued by tableBASE and are identified by codes in the tableBASE command area's ERROR-CODE and subcode fields.

Level 1000 error codes

Some system error messages have been upgraded (by adding '1000' to the error code) in order to reflect a specific VTS environmental problem. For example, 1072 – VTS access failed because the specified VTS is not available.

For more information on tableBASE messages and error codes, see Appendix E on page 407.

7 Customizing tableBASE options

In addition to the variety of facilities embodied in the command set, tableBASE also provides facilities for you to tailor its execution to your particular needs. It does this through execution-time parameters and switches. In this chapter we will discuss these parameters and switches.

The tableBASE software includes delivered default values for these parameters and switches assembled within certain of the tableBASE modules. You can modify these parameters and switches in either of two ways:

- Modify them within the modules supplied and then re-assemble and link the modules. They would then become the default parameters and switches for your company's use of tableBASE (installation defaults). Details of modifying the parameters and switches in this way will be found in the *tableBASE Installation Guide*.
- Use a parameter input file to your programs. In this file, which goes under the DDNAME of TBOPT, you can override any or all of the default installation parameters.

The focus of this chapter is on the parameters and switches themselves, rather than on the syntax of specifying them.

Parameters fall into two groups—those that apply in all tableBASE environments and those that apply only in specific environments.

Note: For a full discussion on tableBASE parameters, see Appendix C on page 389.

Parameters for all tableBASE environments

Most parameters apply to all tableBASE environments. Two parameters that are often modified by users are:

- Table Space Region (TSR) size
- strobe interval

TSR size

tableBASE manages memory in a Data Space referred to as a TSR. It manages memory based on total table demand patterns and attempts to provide optimum memory utilization based on the needs of applications currently running.

The TSRSIZE parameter is an integer that represents the amount of storage to be used for the TSR. This parameter controls the size of the Data Space. The maximum value is 2G. This parameter must be selected with care as choosing a TSR that is too small may mean that your application may run out of space in which to load tables. The TSR must be of size large enough to hold all the tables that are open simultaneously. The delivered default value for the TSRSIZE is 10M for all interfaces.

Strobe interval

tableBASE can produce a report summarizing your use of tableBASE. This report, known as the Table Space Report, includes such information as the space occupied by each table in the TSR, the number of accesses for each table, etc.

You control the frequency with which this information is collected using the STROBE parameter. For example, if you specify a strobe interval of 10,000, the Report record creation routine is called after every 10,000 calls to tableBASE. The report can be printed on demand or when tableBASE terminates.

If Table Space Reports are not desired, there are two ways to turn them off, thus avoiding the related overhead computations:

- STROBE=0 may be specified for the STROBE parameter.
- Omitting the TBTSRPT DD from the JCL (doesn't apply to CICS).

STROBE=0 is useful in the CICS environment in which the Table Space Report records are written to the CICS log file; it will eliminate logging of report records if these are not required.

Parameters that apply to batch only

tableBASE always runs in multitasking mode in all environments except batch. The MULTITASKING parameter may only be specified in batch and it defaults to N. This provides the best performance for single-task batch jobs. For multitasking batch jobs, set MULTITASKING=Y. Performance is slightly slower because tableBASE must perform some internal locking to prevent simultaneous table updates.

Parameters that apply to CICS only

You can specify the CICS Journal file onto which log messages and the Table Space Report records are to be written by the system. This is done via the parameter JFILEID. The default value for this parameter is 99.

Parameters that apply to VTS only

These parameters apply only if you have licensed the VTS Agent. If you have not licensed the optional VTS component , do not change the defaults supplied for these parameters.

There are two parameters that you can specify: VTSFIRST and VTSLAST. The parameters VTSFIRST and VTSLAST control the sequence in which tableBASE searches libraries when VTS is installed.

VTSFIRST is the name of a VTS-TSR that is to be searched for tables before searching any libraries listed in the tableBASE library list (LIB-LIST). To override the installation default for VTSFIRST use the TBOPT DD statement.

VTSLAST is the name of a VTS-TSR that is to be searched for tables after searching any libraries listed in the tableBASE LIB-LIST.

Note: . VTS users who run applications that employ tableBASE's TBCALLV and TBASEV interfaces may also need to override their default SWITCHES. For more details, see "VTS User access" on page 239.

Other parameters

The Date-Sensitive Processing Found Code and Fetch Generic delimiter parameters can impact the execution of certain tableBASE commands.

Date-Sensitive Processing Found Code

The DATERTNX parameter controls the value placed in the Found Code for a Fetch by Count operation using Date Sensitive Processing.

With a Fetch by Count operation it is possible that the row retrieved is not within the date range. When this occurs, the Found Code should be set to N to reflect the logical condition that the retrieved row is not a match.

When Date Sensitive Processing was first introduced as a user contributed program, the Found Code returned under this condition was X. The default setting of N for DATERTNX ensures that this release operates as before and will return an X. If your applications do not require this backwards compatibility, it is strongly recommended that you set DATERTNX to Y so that the found code will be N when this condition occurs.

Fetch Generic Delimiter

This parameter identifies the character to denote the end of the high order part of the key used in the Fetch Generic command.

The default value for the Fetch Generic Delimiter is an asterisk (*****).

For example, `FG,table-name,key*`.

For more information see "Fetch Generic (FG)" on page 91.

8 tableBASE driver command processor for CICS

Introduction

The tableBASE driver command processor (DK1TDRVC, named TBDRIVC in Version 5) is a CICS conversational transaction (normally installed as TBDR) that accepts tableBASE commands from the terminal and invokes the TBLBASE API to execute them. It allows interactive access to all tableBASE operations. It is intended for knowledgeable users performing system administration or application development and testing.

DK1TDRVC accepts commands from and delivers output to the terminal. The top level of DK1TDRVC is simply an input-process-output loop that:

- reads a command
- attempts to execute it
- reports the results.

This loop repeats until DK1TDRVC terminates. The <PF3> key and the command "/*" both give control back to the caller—either the CICS command interpreter or the tablesONLINE/CICS application. The CLEAR key returns control to the CICS command interpreter.

DK1TDRVC can be invoked in any CICS environment as a separate transaction. The default transaction identifier is TBDR. Check with your system administrator to see if this has been changed to match local conventions or to avoid conflict with other names in use at your installation.

DK1TDRVC can also be invoked from tablesONLINE/CICS if your installation has licensed tablesONLINE/CICS.

Security note

DK1TDRVC has been designed for knowledgeable users with a real need for low-level access to the tableBASE system. Where choices had to be made between restricting that access and allowing greater flexibility, flexibility was chosen on the assumption that it would not be abused. System administrators should therefore restrict access to DK1TDRVC, and users should exercise caution in its use.

DK1TDRVC environment

This section describes the DK1TDRVC operating environment.

Abend Status

Before entering the main loop, DK1TDRVC issues a CS command to change TBLBASE Abend Status to N, turning off abend processing. This setting causes TBLBASE to return error codes instead of abending. DK1TDRVC incorporates the returned codes into its report of command results and then accepts the next command as usual. Users may, of course, re-enable abends in TBLBASE with another CS command.

Libraries accessed

When invoked directly as a transaction, DK1TDRVC does not perform any ML commands to set up its library search list (LIB-LIST). Therefore, the site default LIB-LIST is used as the initial library search list.

When invoked by tablesONLINE/CICS, DK1TDRVC picks up the list of concatenated libraries from the current tablesONLINE/CICS list for the user.

In either case, if you wish to work from a library (or VTS subsystem) not on the initial list, your first command should be an ML command to point tableBASE to that library. The current library search list can be checked at any time with the LL command.

Initial settings of the TBLBASE command area

When invoked directly as a transaction, DK1TDRVC initializes the TBLBASE command area with spaces or low-values, as appropriate.

When invoked by tablesONLINE/CICS, DK1TDRVC initializes the command area using data supplied by tablesONLINE; specifically, DK1TDRVC will display the name of the last table, the command, and the error code processed by tablesONLINE. This facility is particularly useful when debugging exit programs. If you want to set up tablesONLINE/CICS to exit into DK1TDRVC whenever the Clear Screen key is pressed, please see the section on Menu Tables in the *tablesONLINE/CICS User's Guide*.

DK1TDRVC sign-on screen

The first screen displayed by DK1TDRVC is shown below (see Figure 8-1). The screen serves two purposes: command entry, and help. It can be redisplayed at any time by pressing <PF1> with the command field (CMD) blank.

```
CMD:    TABLE:          FOUND N I:   A:   ERR:    0 COUNT:        0 L:
 ROW SZ:     0 KEY SZ:   0 FUNC ID:  0 FUNC PARM:          GEN:   0 ERRSUBCD:  0
 1:
 2:
 3:
 HELP:    Enter a Command!              (Use "HP" for general HELP information.)

....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * tableBASE COMMANDS SUPPORTED                    * PF1  HELP AND FORMAT  *
    * Retrieval:  SK, FK, FN, FC, FG, GF, GL, GN, GP. * PF2  DSPL: BROWSE/EDIT*
    * Updating:       IK, RK, DK, IC, RC, DC, MT.     * PF3  ENDS DRIVER      *
    * Table control:  OR, OW, ST, CL, DT, CD, GD, CN, * PF4  >D - ROW TO DSPLY*
    *                 DV, DW, IA, CA, RL, RF.         * PF5  >I - DSPLY TO ROW*
    * Library maint:  DG, XT, RN, NX, CG, LD.         * PF6  >N - >I & RC, GN *
    * System:         LL, ML, LS, CS, BN, SI, DE, LT. * PF7  GP - GET PREVIOUS*
    * DRIVER COMMANDS * * * * * CONVERSION COMMANDS * * PF8  GN - GET NEXT    *
    * DR List Tables in Lib.  * >D / >I to Dspl / Row * PF9  WINDOW (N, +, -) *
    * PR List Rows in Table   * >F Capture Fld Xlation* PF10                   *
    * LG List Rows using Key  * >N Dspl to Row & RC,GN* PF11                   *
    * GH Get-hex     (GN, >D) * >T Hex to Tablename   * PF12                   *
    * RH Replace-hex (>N, >D) * >L Hex to Lock field  *                        *
    * SS Set Subsystem        * >H Tblname & LL to Hex* CLEAR RETURN TO CICS  *
    *                         * >S LT name to TABLE   * PA? HELP/FORMAT/TBONLY*
    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Figure 8-1: DK1TDRVC sign-on help screen

DK1TDRVC input area

The first seven lines, (the ruler, and everything above it) are common to all DK1TDRVC screens (see Figure 8-2).

```
CMD:    TABLE:          FOUND:   I:   A:   ERR:    0 COUNT:        0 L:
ROW SZ:      0 KEY SZ:   0 FUNC ID:  0 FUNC PARM:          GEN:   0 ERRSUBCD:  0
1:
2:
3:
HELP:    Enter a Command!              (Use "HP" for general HELP information.)
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
```

Figure 8-2: DK1TDRVC input area

The command lines

The first two lines in Figure 8-2 are known as the command lines. They provide an area labelled “CMD:” for typing the next command and several other fields. These fields map directly into the command area of a TBLBASE call, the fixed area common to all such calls.

TABLE

The "TABLE" field on the first command line names the table being operated on for all table-specific commands. It is initialized to spaces on entry to DK1TDRVC directly from CICS or to the last table accessed by tablesONLINE/CICS if DK1TDRVC is called from tablesONLINE/CICS. Once set, it retains its value until altered in one of the following ways:

- by changing DK1TDRVC modes which resets this field along with all others on the command line to their initial value (see “Special functions” on page 252 for information on modes)
- by explicit data entry to the field
- by the NX command, which places the next table name from a library into the field
- by the DR command, which displays a directory of a library and leaves the last table name displayed in the field
- by the CN command, which changes the name of a table in memory and leaves the new name in the field
- by the RN command, which changes the name of a table on a library and leaves the new name in this field
- by entering data on line one (1:) in hexadecimal and converting it to a table name with the >T command

The last method allows entry of table names containing special characters, such as tablesONLINE/CICS controlling tables.

FOUND, ERR, and ERRSUBCD

"FOUND" and "ERR" are protected fields that report the results of the last command executed. "FOUND" will be Y if a specified row was found, otherwise N. "ERR" and "ERRSUBCD" are set to zero if the command executed successfully, otherwise they are set to some non-zero error code. To interpret these codes, look for the related error message on the sixth screen-line, known as the message line (see “tableBASE error codes” on page 408.)

I Field

The "I" field on the command line is a dual purpose field. It can be used to invoke an indirect open or to convert table names.

If you set the field to I or a space, this value is passed to the TBLBASE API as the INDIRECT INDICATOR field. See "Open for Write (OW)" on page 123, "Open for Read (OR)" on page 121, and "Open Indirect" on page 61.

This field may also be set to V=view. These values are not passed to the TBLBASE API (a space is substituted), but are used by DK1TDRVC to rename tables.

A value of V=view in the field indicates that the table name should be converted to a View name (bit six of its first character set to a value of zero, counting the least significant bit as bit zero) before use.

Once set, the "I" field retains its value so that a series of operations on Views can be performed without having to reset it. The NX command, which gets the next table name from the tableBASE library directory, sets this field to the appropriate type for that table, while the DR command, which gives a directory of the library, sets the field to the type of the last table listed.

A Field

The "A" field is used to override the Abend Status. If the field is set to Y, the transaction will abend if an error occurs while executing the tableBASE command. If the field is set to N, the transaction returns an error code if an error occurs when calling tableBASE. If the value is left blank the Abend/Error Status in effect is the last value used in the Change Status Command.

COUNT

The "COUNT" field reports the table position of the row retrieved or saved for those commands that change the COUNT field of the COMMAND-AREA. "COUNT" may also affect the next command executed if that command takes a count parameter, for example, the FC (Fetch by Count) or GN (Get Next row) commands. "COUNT" is therefore an unprotected field that users may set to control the operation of such commands.

Note: In a multitasking or multi-user online environment, results for commands that utilize a count or sequential accessing of table rows may get unexpected results if another task/user is updating or deleting rows in the same table.

LOCK field

The LOCK-LATCH field of the command line, labelled "L", holds the eight-character password required to control table updating. This is used to limit table update capability to a particular transaction, user, or terminal.

DK1TDRVC allows data entry to this field, either directly or in hexadecimal on line one (1:), which is then translated to the field with the >L command (see below), giving the user update ability to any table for which the lock is known.

CAUTION: Cautious use of this mechanism is strongly encouraged—bypassing the system's controls on multiuser write access is rarely necessary.

ROW SZ

This field denotes the relevant portion of DK1TDRVC's row area to be used for retrieval and update requests. If this field is zero, then the actual row length is used, as specified in the table definition.

KEY SZ

A non-zero value in this field denotes the relevant length of a generic key parameter in a Fetch Generic request. If this field is zero, and the command is FG, then the key parameter (which, as with all command parameters is entered on line 1:) is assumed to contain a delimiter (usually an asterisk (*****)) that marks the end of the generic key.

FUNC ID

This field is used to distinguish between normal tableBASE processing and other specialized types of tableBASE processing. Date-Sensitive Processing is the only special processing available at this time.

FUNCTION-ID = 0 for normal processing

FUNCTION-ID = 1 for Date-Sensitive Processing

FUNC PARM

This field is used for Date-Sensitive Processing. This date and the effective date and expiry date stored on a date-sensitive table must be of the same format: YYYYMMDD. If this field is spaces or low values, then the system's current date will be inserted here by TBLBASE and the current date will be used for Date-Sensitive Processing.

GEN

This field returns the generation number of the table.

ERRSUBCD

This field provides more detailed information on the error code. Not all error codes will have subcodes. See “tableBASE error codes” on page 408 for more details.

Parameter lines 1:, 2:, and 3:

Referring to Figure 8-2 , lines 1:, 2:, and 3: are found after the command line. These have several functions in DK1TDRVC. Commands requiring parameters beyond those on the command line will look for them here, delimited by commas. The > commands that convert data between formats will expect to find their format instructions on line 3: and to find or place data on lines 1: and 2:.

Message line

The sixth screen line is known as the message line. It may contain error messages, help information, instructions or suggestions for the user at various points in the program.

When a tableBASE call returns an error status, TBERROR appears at the left of the message line. Errors detected in the DK1TDRVC conversion routines cause ERROR to appear at the left of the message line. If DK1TDRVC is in HELP or FORMAT mode (see Ruler line), an error message will appear on the right side of the message line. Error codes for tableBASE errors appear in the ERR" and "ERRSUBCD" fields on the command line, while messages for those errors appear on the message line.

Ruler line

The seventh screen line, the last of those common to every DK1TDRVC screen, is a ruler line to assist in finding character offsets during editing. See Figure 8-2.

Below the ruler, the first screen displayed lists the various commands available under DK1TDRVC in three groups on the left and gives the special keys recognized by the program down the right side.

Special functions

The right hand area of the main screen lists PF keys and other special keys (see Figure 8-3).

```
* * * * * * * * * * * * * *
* PF1  HELP and FORMAT    *  <- effect depends on mode HELP/FORMAT/TBONLY
* PF2  DSPL: BROWSE/EDIT  *
* PF3  ENDS Driver        *  <- return to caller, e.g. tablesONLINE/CICS
* PF4  >D - Row to Dsply  *
* PF5  >I - Dsply to Row  *
* PF6  >N - >I & RC, GN   *
* PF7  GP - Get Previous  *
* PF8  GN - Get Next      *
* PF9  WINDOW (+, - N)    *
* PF10                    *  <- not used in this release
* PF11                    *  <- not used in this release
* PF12                    *  <- not used in this release
*                         *
* CLEAR RETURN to CICS    *  <- bypassing caller
* PA? HELP/FORMAT/TBONLY  *  <- switch mode to next of HELP/FORMAT/TBONLY
* * * * * * * * * * * * * *
```

Figure 8-3: Special function key description

Mode Switch – PA2/PA1

The Mode Switch controls the degree of help that may appear on the screen. There are three settings: HELP, FORMAT, and TBONLY. The PA2/PA1 key moves you to the next of these modes in a ringed sequence, from HELP to FORMAT to TBONLY, and back to HELP. The mode is displayed on the message line described above.

In the default HELP mode, entering the HP command or pressing <PF1> displays all available help. A summary of the relevant command's syntax appears on the message line and additional help text appears below the ruler, overwriting any currently displayed information. In FORMAT mode, below-the-ruler help text is suppressed to preserve the display. In TBONLY mode, all help is suppressed to avoid table accesses not explicitly asked for by the user so that the state of an application program can be mimicked as exactly as possible for debugging. In TBONLY mode, the HP command is inoperative while the <PF1> key displays the command summary similar to Figure 8-1 used for the sign on screen. This is the only help available in TBONLY mode.

If it is essential to invoke DK1TDRVC so that no commands are invoked during initialization except the ones entered on the command line, the TBONLY mode can be entered directly. Enter "TBDR TBONLY" on the CICS screen to initiate the transaction.

The leftmost word on the message line of the DK1TDRVC screens indicates which of DK1TDRVC's three modes is in use; HELP, FORMAT, or TBONLY. Modes are switched with the PA2 key; the difference among modes is the amount of help available to the user.

When switching modes, DK1TDRVC re-initializes itself by:

1. Resetting all fields of the command line to their state when the program was started
2. Clearing the message area
3. Setting abend status off
4. Returning the library concatenation list and Command Area to its original state if invoked from tablesONLINE/CICS
5. Getting a new system date.

It does not change the EDIT switch (see below), nor does it automatically perform any ML commands to ensure that tableBASE and DK1TDRVC have consistent library information. If libraries other than the defaults are in use, the user should issue the appropriate MLs after each mode switch.

DK1TDRVC is written to respond to any PA key as a signal to switch to the next mode, but some environments trap PA1 for their own purposes and do not pass it to applications, and not all terminals provide PA3. So, if one PA key does not work, try another.

Help <PF1>

Entering the name of any command in the command area and pressing <PF1> provides help on that command, if you are in the default HELP mode. In FORMAT mode, it displays only the command argument format information on the message line.

You may obtain a list of available commands by entering "??" as a command. Or, you may enter "?K" for a display of PF Key functions.

This is the screen obtained by entering HP, HELP, and pressing <PF1> or <Enter> (see Figure 8-3).

```
CMD: HP TABLE:          FOUND N I:   A:   ERR:    0 COUNT:        0 L:
ROW SZ:  1292 KEY SZ:   0 FUNC ID:  0 FUNC PARM:          GEN:   1 ERRSUBCD:  0
1:
2:
3:
HELP:    Use PF1 with Command for each command.       (Set CMD=spaces for list)
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
   HELP will provide Format information above the ruler and detailed help
   below the ruler (clearing the display).  In FORMAT mode, only format
   information is displayed thus preserving the display below.

   Enter '??' for a list of Command Description; '?K' for help on PF keys

   NOTE: Using ENTER without entering a command will repeat or continue the
         previous command.  For example: Using ENTER with PR or SPACES prints
         the contents of the TABLE from the beginning, subsequent ENTER's
         continue from the current Count.  Some commands are NOT tableBASE
         commands but enhance the use of the Driver. (i.e. HP, DR, PR, LG)

   The purpose of the tableBASE driver is to simplify user testing providing
   'hands on' testing and demonstration of tableBASE commands.  The Driver is
   intended to be an Educational aid and a Maintenance tool for tableBASE.

   *** Press Enter to continue with TUTORIAL.
```

Figure 8-4: DK1TDRVC help screen

One way to obtain help is to enter the HP command then page through the available help material. For more specific help, enter a command name in the command field of the first line, then press <PF1>.

<PF1> with any command in TBONLY mode or <PF1> with a blank command area in HELP mode re-displays the DK1TDRVC sign on screen.

Browse or edit <PF2>

<PF2> toggles DK1TDRVC between browse and edit modes, the setting on entry being browse. Use browse mode whenever possible to prevent accidental update of table content.

Data conversion <PF4>, <PF5>, <PF6>

For information on the >D, >I, and >N functions see the section below on conversion commands.

Traversing tables <PF7>, <PF8>

<PF7> and <PF8> invoke the Get Previous table row and Get Next table row commands. Use them to walk through a table with a minimum of key strokes.

Leaving DK1TDRVC

The CLEAR key, the "/*" command, or <PF3> all terminate DK1TDRVC. The difference is that CLEAR returns you directly to CICS command level, while <PF3> returns you to the program which called DK1TDRVC, typically tablesONLINE/CICS.

Edit switch

The EDIT Switch, toggled by <PF2>, controls whether the actual row data displayed on the lower part of the DK1TDRVC screen may be edited. When this switch is set to EDIT, data is transferred directly back and forth between the screen and the table row in memory. This is convenient for rows containing only displayable characters, but definitely not a good idea for rows with binary data.

When this switch is OFF, all non-display characters are converted to periods, which can be safely displayed, and direct editing of the row data is disabled. Conversion commands are available to format data for editing in the upper-screen area and to move edited data back to the row. This is more work (though extra commands are provided to simplify common sequences), but both safer and more powerful than the direct edit. It is the default mode.

DK1TDRVC commands

Once abend processing is appropriately set and a library located, subsequent commands are entirely at the user's discretion. Commands are executed in sequence as they are read from the terminal. A report of results is printed after each one until END, <PF3>, CLEAR, or "/*" terminates the session.

A DK1TDRVC command begins with a two-character command name and may also contain one or more parameters to be passed to the command. The first two lines of the DK1TDRVC screen are command lines with one field for each part of the TBLBASE command area. Additional parameters, for calls requiring them, are given on subsequent screen lines, delimited by commas. Where a parameter is not given—either fewer fields than the required number of parameters, or some field is null (two delimiters with nothing between them)—then spaces are passed to the TBLBASE API causing the tableBASE default for the field to be used. Parameters shorter than the length available for them in the TBLBASE command area are left-justified and padded with spaces.

Two features of the DK1TDRVC command interface are worth noting before discussion of specific commands:

- If, after a command executes, the user presses the <Enter> key again, the command is re-executed. This is particularly useful for iterative commands such as GN (Get Next row) or FG (Fetch rows matching a Generic key). The first iteration sets the count to the row retrieved and subsequent iterations use and reset that count, resulting in scrolling through the table.
- A blank command area has an effect. Pressing <Enter> with no command executes the PR command, displaying the next seventeen table rows. Pressing <PF1> redisplays the first DK1TDRVC screen with its list of available commands.

Complete listing of DK1TDRVC commands

Entering "??" in the command area and pressing <Enter> or <PF1> gives you a list of commands supported in DK1TDRVC. The list spreads over several screens; to scroll through the screens continue pressing <Enter>. To leave the list, enter another command in the command area.

All commands are listed in Table 8-1. Some commands apply only to DK1TDRVC and are noted as Driver Commands.

Table 8-1: DK1TDRVC command summary

Command	Description	Note
**	Comment	
AB	ABend with 0C7 (used to force dump)	Driver command
BN	display signon BaNner	tableBASE command
CA	Create Alternative table definition	"
CD	Change table Definition	"
CG	Change number of Generations retained in library	"
CL	CLose table	"
CN	Change table Name in region	"
CS	Change Status switches	"
DC	Delete by Count	"
DG	Delete a table Generation	"
DK	Delete by Key	"
DR	list tables in library (uses NX and GD)	Driver command
DT	Define Table	tableBASE command
DV	Divert table to another library (where it does not exist)	"
DW	Divert table to another library (where it exists)	"
FC	Fetch by Count	"
FG	Fetch Generic	"
FK	Fetch by Key	"
FN	Fetch Next by key	"
GD	Get table Definition	"
GF	Get First table row	"
GH	Fetch by Count & Translate Row into Hex on Lines 1-3	Driver command
GL	Get Last table row	tableBASE command
GN	Get Next table row	"
GP	Get Previous table row	"

Table 8-1: DK1TDRVC command summary (Continued)

Command	Description	Note
IA	Invoke Alternate Index definition	"
IC	Insert by Count	"
IK	Insert by Key	"
LD	List Directory	"
LG	List Generic (uses repeated FG)	Driver command
LL	List Library search order	tableBASE command
LS	List Status switches	"
LT	List open Tables	"
LV	List VTS settings	"
ML	Modify Library search order	"
MT	EMpty table	"
NX	return NeXt tablename in library	"
OR	Open table for Read	"
OW	Open table for Write	"
PR	List Rows in table (default command for no command)	Driver command
QR	open view for Read	Driver command
QW	open view for Write	Driver command
RC	Replace by Count	tableBASE command
RH	Translate Hex into Row, Replace Row, Get Next into Hex	Driver command
RK	Replace by Key	tableBASE command
RL	ReLease a table	"
RN	Rename table Name in library	"
SI	Set Indirect table search parameter	"
SK	Search by Key	"
SP	Special Processing	Driver command
SS	Select Subsystem	"

Table 8-1: DK1TDRVC command summary (Continued)

Command	Description	Note
ST	Store Table	tableBASE command
VE	Show version of DK1TDRVC	Driver command
XT	eliminate all Table generations from library	tableBASE command
>D	Translate row Data into lines 1-3	Driver command
>F	Move current View row into Driver Translate Control	"
>H	Translate tablename & lock key into hex on line 1	"
>I	Translate lines 1-3 into row	"
>L	Translate hex into Lock key	Driver command
>N	Translate lines 1-3 into row and Get Next row	Driver command
>S	Allows transfer of a hex-character table name from the LT command output to the table name field so another command can be invoked against it.	"
>T	Translate hex into Tablename	"

tableBASE commands

Most of the commands available are those supported by the tableBASE API TBLBASE (see Figure 8-5). The names of these commands, the order of parameters for each command, the size limit and default value for each parameter, and the error codes used are all determined by the TBLBASE API. See Chapter 3 “tableBASE commands” on page 57 for details of these commands and their parameters.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* tableBASE COMMANDS SUPPORTED                           *
* Retrieval:  SK, FK, FN, FC, FG, GF, GL, GN, GP. *
* Updating:       IK, RK, DK, IC, RC, DC, MT.      *
* Table control:  OR, OW, ST, CL, DT, CD, GD, CN, *
*                 DV, DW, IA, CA, RL, RF.          *
* Library maint:  DG, XT, RN, NX, CG, LD.          *
* System:         LL, ML, LS, CS, BN, SI, DE, LT. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Figure 8-5: DK1TDRVC tableBASE commands

For each of the commands DK1TDRVC does the following:

1. Puts data from its command line into a parameter block for the TBLBASE API
2. Takes row data, if required, from below the ruler
3. Parses any data on input lines 1-3 into comma separated fields and places the results in additional TBLBASE parameter fields, if the command requires these
4. Fills null fields with default values
5. Calls TBLBASE API to execute the command
6. Reports results.

The "ERR", "FOUND", and "COUNT" fields on the command line indicate command results. In HELP or FORMAT modes, the ERR code is also interpreted to give a message on the message line. Any row fetched appears below the ruler.

DRIVER commands

For user convenience, a few additional commands are provided under DK1TDRVC. These are listed on the first screen as DRIVER COMMANDS (see Figure 8-6). Most of these commands are macros, sequences of common commands. Each sequence has been assigned a name so it can be executed more easily.

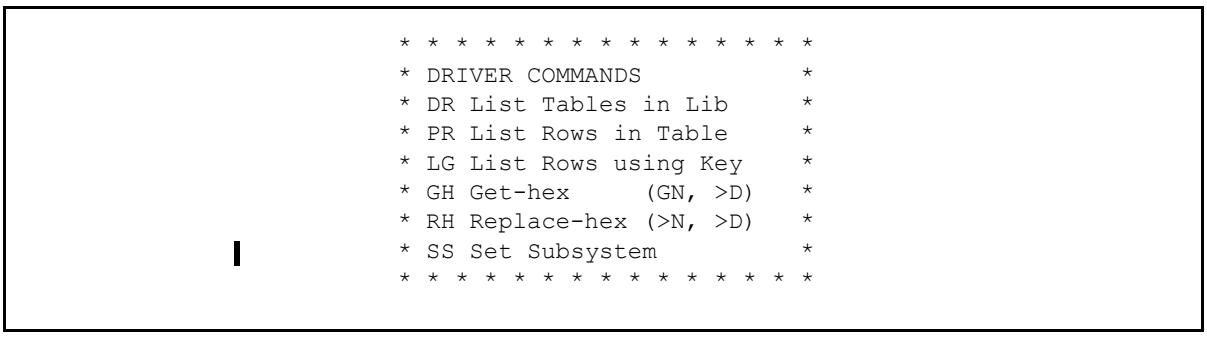

Figure 8-6: DK1TDRVC driver commands

All of these commands use tableBASE calls and leave the FOUND, COUNT, and ERROR fields on the command line as set by the last tableBASE call executed. All of them exit and return control to the user after any call which returns an error status.

Print (PR)

Use the Print (PR) command to fill the screen below the ruler with table rows by executing a series of GN (Get Next row) calls. It leaves COUNT set to the last row retrieved so just pressing <Enter> to re-execute the command allows for forward scrolling through the entire table. <PF8> has the same effect, and <PF7> scrolls backward through the table. Explicitly re-entering PR in the command field instead of pressing <Enter> will reset the count and restart scrolling at the beginning of the table (see Figure 8-7) below for a sample of the information displayed after executing the PR command.

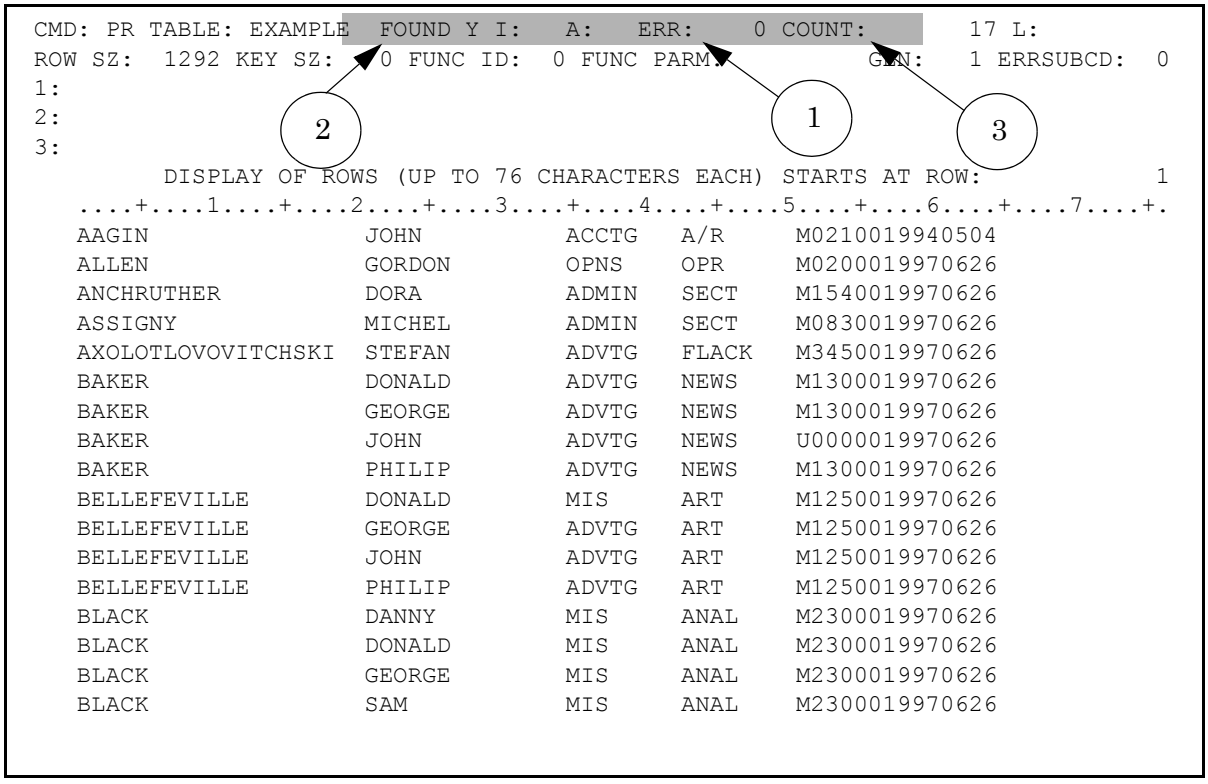

Figure 8-7: DK1TDRVC listing table data

Seventeen rows are displayed, one to a line. Only the first 76 characters of each row are visible.

The command line reports that:

1. the command worked (ERROR is zero) (Figure 8-7)
2. the last retrieval succeeded (FOUND is Y) (Figure 8-7)
3. the last row retrieved was number seventeen (COUNT is 17) (Figure 8-7).

If the end of the table had been reached, all rows successfully retrieved would have been displayed, plus an end of data marker. FOUND, COUNT, and ERROR would have been as set by the last tableBASE GN command executed, the one encountering the end of the table. The message line provides a prompt to assist in interpreting the display.

For this display (Figure 8-7):

- When an override length is entered in ROW SZ for retrieval or update commands, it will be used to do the retrieval or update, but the ROW SZ field will always display the actual row length after the command. Even though the actual row length is displayed in this field, the override length originally entered in this field will continue to be used for all commands until another input is entered in the ROW SZ field. The messages

  ```
  OVERRIDE LENGTH xxxxx BYTES DISPLAYED
  ```



```
OVERRIDE LENGTH xxxxx BYTES INSERTED
```
will be displayed after the update or retrieval command, if an override length less than the row size was used for the retrieval or update.

- For tables with row sizes greater than 1292 bytes, the online driver can only display up to 1292 bytes. The message
```
MAXIMUM 1292 BYTES DISPLAYED
```
is displayed for such retrievals to indicate this.
- For tables with key sizes more than 228 bytes, you need to use the area below the ruler to insert or replace the rows as the area above the ruler can only accomodate up to 228 bytes of data. To do this, press the F2 key and enter the row data below the ruler instead of above the ruler. To toggle back to normal mode, hit the F2 key again.

A variety of options are available for the next display. Press <Enter> to repeat the PR command and request a display starting at row eighteen, one beyond the current count. Updating the COUNT field, for example, to 50 and pressing <Enter> requests a display starting at row 51, one beyond that count. Updating the command field with PR restarts the display at row one.

List Generic (LG)

The LG or List Generic command does much the same thing, but more selectively. It takes a partial key, for example "AB*", and uses the tableBASE FG (Fetch Generic) command to fetch all rows matching that key (all rows with keys beginning with "AB"). The same result would have been obtained by entering a 2 in the "KEY LENGTH:" field and AB for a partial key. See "Fetch Generic (FG)" on page 91 for additional information.

Scrolling works as described for PR.

Note: The DR or Directory command provides a similar display, but it uses the tableBASE NX command to fetch successive table names from the tableBASE Library Index, rather than rows from a table. The message line shows the number of blocks allocated to and available on the library. The <Enter> key may be used to scroll forward through the list of table names.

List Directory (LD)

The List Directory (LD) command is also available for generating directory lists in a temporary table. Scrolling with LD works as described for PR. For additional information, see "List Directory (LD)" on page 107.

Get Hex (GH)

Two DK1TDRVC commands Get Hex (GH) and Replace Hex (RH) facilitate hexadecimal or other unusual-format editing. The GH command fetches a row by count (default is next row) and displays it both in row format and in whatever format has been specified on the third display line (default is characters interpreted in hex). A single command will pick up the next row and have it displayed in the format needed for editing.

Replace Hex (RH)

The Replace Hex (RH) command has a related function but stores the current row before picking up the next. The RH command first translates data on the formatted display lines into the row area, then replaces the row in the table, gets the next row, and finally displays that in the desired format.

Set Subsystem (SS)

The Set Subsystem (SS) command sets the VTS subsystem name for subsequent tableBASE calls. All table access requests will be satisfied from the named VTS-TSR from this point on, until another SS command resets the subsystem name. This is a Driver-specific command; although it is not necessary in application programs using TBLBASE, it is required in the Driver to provide access to the TB-PARM parameter of TBLBASE calls.

Issuing an SS command with a blank subsystem name will reset subsequent tableBASE API calls to use the local TSR.

Data conversion commands

DK1TDRVC also has built-in facilities for translating data from one format to another to allow editing of data in formats more palatable than the storage format. You might, for example, need to edit character fields containing byte values not readily displayed or manipulated except as hexadecimal values, or want a number displayed in some numeric format other than that given in the View. The DK1TDRVC screen provides a display area for such translated data, as well as for the original format row display. The first screen lists available commands of this type as CONVERSION COMMANDS (see Figure 8-8). They all have two-character names beginning with the ">" character.

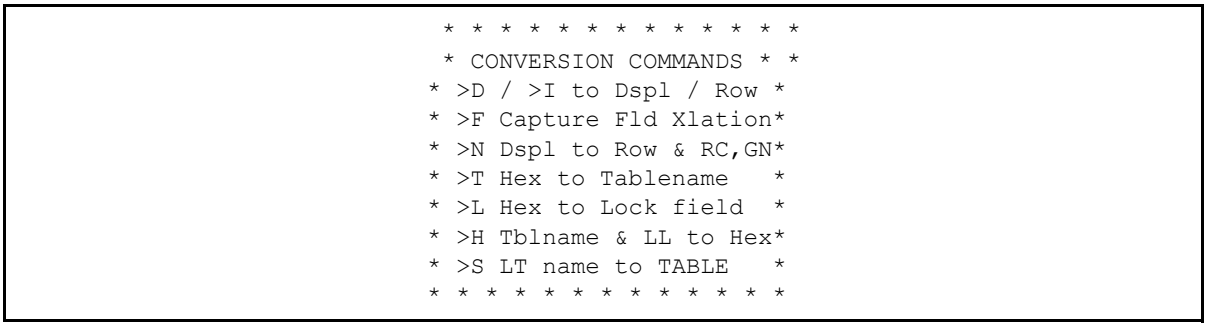

Figure 8-8: DK1TDRVC data conversion commands

We will first discuss the set of commands (>T, >L, >H) which deal with the TABLE and LOCK fields in the command line. Then we will discuss the commands (>D, >I, >N, >F) that are used when editing Data Tables containing non-displayable characters.

Tables containing only displayable characters can be edited directly by turning the Edit switch on with <PF2> and entering data directly into the row display area below the ruler on the DK1TDRVC screen.

For other types of data, conversions between the storage format and some editable format must be applied. This is usually done with the Edit switch off. The area below the ruler is then a protected area showing a displayable approximation to the storage data (non-display characters shown as periods) while the area above the ruler shows the editable formatted version.

>T and >L commands

The >T and >L commands both expect hexadecimal data on line 1: of the input area, which will be converted to characters and placed on the DK1TDRVC command line. The >T command puts data in the command line TABLE field; >L in the L (LOCK) field. These commands allow access to tables whose names contain non-display or hexadecimal characters. They also enable you to deal with non-display or hexadecimal characters in the lock field. The hexadecimal data on line 1: must be 16 characters padded with spaces (hexadecimal '40').

>H command

The >H command performs exactly the opposite function and converts both the tablename and the lock into hex on line 1:.

>D, >I and >N commands

The >D command converts row data from below the ruler into a display format and displays the results above the ruler.

The >I command provides the inverse operation, moving data into the row.

The >N command converts above-the-ruler data, putting results into the row area below the ruler, replaces that row in the table, and gets the next row. Each of these commands is also available using a PF key. Table 8-2 lists the Command to PF key conversion.

Table 8-2: PF Keys for data conversion commands

Command	PF Key
>D	<PF4>
>I	<PF5>
>N	<PF6>

>S command

The >S command provides a mechanism to transfer a table name containing hex characters from the LT command output to the table name field so another command (GD or CL, for example) can be invoked against it. Preceded by the LT command, >S requires that line 1 contain the REF# for the table identified in the LT command output. The command will copy the table identified by this REF# to the "TABLE:" field on the command line.

For example, the LT command provides a list of tables in the TSR:

```
 CMD: LT TABLE:          FOUND N I:   A:   ERR:    0 COUNT:        0 L:
 ROW SZ:     0 KEY SZ:   0 FUNC ID:  0 FUNC PARM:          GEN:   0 ERRSUBCD:  0
 1:                  ----- HIGH WATER MARKS -----         - CURRENT SPACE -
 2:     TSR           OPEN        TSR         TSR          OPEN        TSR
 3:   ENTRIES         TABLES      USAGE       SIZE         TABLES      USAGE
          30          1,468K     1,656K   102,400K         1,048K     1,236K
    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
    REF    NAME   W/R L/V IA?      CALLS      SPACE      ROWS RWS-BF-EXP (F11)>>
    ------------------------------------------------------------------------------
      16 TBOLPROF  R   L   N          69     16,384        24        32
      17 TABLE003  W   L   N           1      8,192        10        57
      18 TBOLHELP  R   L   N           9     12,288        53        70
      19 MS......  W   L   N          42      8,192         0        47
      20 MU......  W   L   N          37      8,192         1       336
      21 LC......  W   L   N         196      8,192         0       336
      22 TESTDESC  R   L   N          74     98,304       417       511
```

Figure 8-9: A list of tables in the TSR, as seen via the LT command

If you want to edit or browse a table in this list, but it contains non-displayable characters, use the **>S** command. In the example below, the table Ref# is entered onto Line 1:

```
 CMD: >S TABLE: MS......  FOUND N I:   A:   ERR:    0 COUNT:        0 L:
ROW SZ:     0 KEY SZ:   0 FUNC ID:  0 FUNC PARM:          GEN:   0 ERRSUBCD:  0
1: 19
2:
3:
         LT LIST NAME COPIED TO "TABLE" FIELD
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
   REF    NAME   W/R L/V IA?      CALLS      SPACE      ROWS RWS-BF-EXP (F11)>>
   ------------------------------------------------------------------------------
     16 TBOLPROF  R   L   N          69     16,384        24        32
     17 TABLE003  W   L   N           1      8,192        10        57
     18 TBOLHELP  R   L   N           9     12,288        53        70
     19 MS......  W   L   N          42      8,192         0        47
     20 MU......  W   L   N          37      8,192         1       336
     21 LC......  W   L   N         196      8,192         0       336
     22 TESTDESC  R   L   N          74     98,304       417       511
```

Figure 8-10: Sample using the >S command

The following messages are possible using the **>S** command:

`LT LIST NAME COPIED TO "TABLE" FIELD`	The LT list was successfully copied to the "TABLE:" field on the command line.
`REF# PROVIDED NOT ON LT LIST`	The reference numbers provided on line 1 do not exist on the LT list.
`LT LIST NOT FOUND`	The LT command was not executed before doing the >S command.
`INVALID REF# ON LINE 1`	The reference number entered on line 1 is not a numeric field.

GH and RH commands

The GH and RH commands also use this conversion facility. GH is equivalent to the sequence GN (Get Next), >D. RH is equivalent to >I, RC (Replace by Count), GN, >D.

Data conversion parameters

The >D, >I and >N commands, the equivalent <PF4>, <PF5>, and <PF6> keys, and the more complex GH and RH commands all expect to find data conversion parameters on line 3: as five comma-separated fields (see Table 8-3). These conversion parameters are retained by DK1TDRVC for subsequent data conversion operations.

Table 8-3: Conversion parameters

Field	Data type	Refers to
DISPLAY FORMAT	character	Display area above the ruler
DISPLAY LENGTH	numeric	length in bytes
STORAGE FORMAT	character	Row data area below the ruler
STORAGE LENGTH	numeric	length in bytes
LOCATION	numeric	location

Units of all numeric fields are bytes. Formats are those used in tablesONLINE/CICS Views. See the *tablesONLINE/CICS User's Guide* for details. The most common conversion is between character (X) and hexadecimal (Y), but any conversion possible in tablesONLINE/CICS is possible here.

Like tablesONLINE/CICS, DK1TDRVC is designed to trap most errors in format instructions or in the data on which they operate and generate messages.

>F command

The >F command, given a field description from a tablesONLINE/CICS View, loads the relevant conversion data into DK1TDRVC. You can then switch to the related Data Table and convert its data by tablesONLINE/CICS rules from within . This procedure is recommended whenever tablesONLINE/CICS Views are available. For example, to access a 100-byte character field at offset 750 in the table row enter this conversion format on line three: 3: Y,200,X,100,750

The >D command then creates a 200-character hexadecimal display above the ruler, taking its data from bytes 750 to 849 of the row. Then the data may be edited and moved back to the row area using the >I command. To combine the conversion with other operations for greater convenience, the >N or RH commands could be used.

Figure 8-11 is an example of the >N command as the command is applied to the table shown in Figure 8-7. The screen sample shown in Figure 8-11 is displayed immediately after <Enter> is pressed.

```
CMD: >N TABLE: EXAMPLE  FOUND Y I:    A:    ERR:     0 COUNT:         2 L:
ROW SZ:  1292 KEY SZ:    0 FUNC ID:  0 FUNC PARM:           GEN:    1 ERRSUBCD:  0
1: 200.00
2:
3: 2,6,N,5,52
TRANSLATE DISPLAY FORMAT=2, LENGTH=006, INTO ROW FORMAT=N, LENGTH=005, AT 0052
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
   ALLEN               GORDON         OPNS    OPR     M0200019970626
```

Figure 8-11: DK1TDRVC >N Data conversion command

The message line shows the translation that was performed (see Table 8-4).

Table 8-4: Message line data

From:			
	DISPLAY FORMAT	2	two decimal places
	DISPLAY LENGTH	6	bytes, including decimal
To:			
	FIELD FORMAT	N	numeric, integer
	FIELD LENGTH	5	bytes
At:			
	LOCATION	52	in row area

The translation picked up the required 6 bytes from the beginning of line one, "200.00" in this case, converted this to the 5-byte numeric string "02000", and stored that beginning at character 52 of the row area, overwriting the existing data.

The >N command processing does not stop with that. The modified row is replaced in the table and the next row is fetched as well. That is the row for "ALLEN GORDON" which is displayed under the ruler line.

Yet another step could have been taken by using the RH, Replace Hex, command which does everything >N does and then converts the new row data, putting some new string in place of the "200.00" on line one.

To perform another similar conversion elsewhere in the row, enter a string such as: `3: ,,,,100` on line three. Fields of the conversion specification left empty in such a string retain their old values, resulting in: `3: 2,6,N,5,100`

The new conversion uses data from a different LOCATION, but otherwise is the same as the previous one.

Date-sensitive processing

Date-sensitive processing in DK1TDRVC is supported through the use of the FUNC ID and FUNC PARM fields in the second command line.

The following sequence of commands in the Driver illustrates the use of these fields in processing date-sensitive tables. The PR (Print Table Contents) may be used to show all the rows in a date-sensitive table (see Figure 8-12).

```
CMD: PR TABLE: TBDATE01 FOUND N I:   A:   ERR:    0 COUNT:       11 L:
 ROW SZ:  1292 KEY SZ:   0 FUNC ID:  0 FUNC PARM:          GEN:   1 ERRSUBCD:  0
 1:
 2:
 3:
          DISPLAY OF ROWS (UP TO 76 CHARACTERS EACH) STARTS AT ROW:          1
    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
    BC        1995010119951231
    BC        1996060119960831....
    BC        1996090119990101....
    MAN       1996123199991231....
    ONT       0000010199991231....
    PQ        0000010119960930....
    PQ        1996100119970901....
    PQ        1997100119980605....
    PQ        1998060699991231....
    SASK      1998010119981231....
    ****************** END OF ROWS *********************
```

Figure 8-12: DK1TDRVC date-sensitive processing example data

The following PR command (see Figure 8-13) shows the date-sensitive table as it appears on January 1, 1998. The Driver is set for FUNCTION ID 1, Date-Sensitive Processing. The FUNCTION PARAMETER supplied is the date in the form YYYYMMDD.

The FUNC ID and FUNC PARM fields remain on the screen until they are explicitly changed by the user. Next, a Fetch by Count (FC) command with a count of 7 is entered. The N found code indicates that although there is a seventh row on the table it is not valid for the date setting (see Figure 8-13). Since no row was found for display, the display below the ruler line remains as it was before the command was executed.

```
 CMD: FC TABLE: TBDATE01 FOUND N I:   A:   ERR:    0 COUNT:        7 L:
 ROW SZ:  1292 KEY SZ:   0 FUNC ID:  1 FUNC PARM: 19980101 GEN:   1 ERRSUBCD:  0
 1:
 2:
 3:
 FUNC: 01

....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
    BC        1996090119990101....
    MAN       1996123199991231....
    ONT       0000010199991231....
    PQ        1997100119980605....
    SASK      1998010119981231....
    ****************** END OF ROWS *********************
```

Figure 8-13: DK1TDRVC FC command using date-sensitive processing

9 TBDRIVER command processor for Batch/TSO

The TBDRIVER (DK1TDRV) program accepts tableBASE-like commands from a terminal or a file and invokes tableBASE to execute them. TBDRIVER also has additional commands (referred to as macros) created to make long tasks easier such as PRINT TABLE (PR).

TBDRIVER is designed for technical users doing administration or application development and testing. It is a convenient utility for checking out tableBASE commands without writing a special test program. It may also be used for making changes or corrections to tables or libraries.

Note: The *tableBASE Administration Guide* provides information on how to refresh tables in a Read/Write VTS-TSR using TBDRIVER.

How to invoke TBDRIVER (DK1TDRV)

The TBDRIVER command may be invoked in a batch job, or interactively through the TSO CLIST TBDR or the tablesONLINE/ISPF menu. TBDRIVER uses the files listed in Table 9-1.

Table 9-1: TBDRIVER files

DDNAME	Optional	Usage
CMD	No	Input command stream (80-byte records)
SYSOUT	No	Display of each input command and the result of its execution
TBSYSLB	Yes	tableBASE library containing TBDRHELP table (required for HP command)
MAINLIB	Yes	tableBASE library used in ML command (other DDNAMES/libraries could be used instead of MAINLIB, with appropriate ML command usage)
TBOPT	Yes	Allows overrides to system parameters.
TBTSRPT	Yes	Table Space Report, if desired

The TSO CLIST TBDR allocates CMD and SYSOUT to the terminal, and allocates a default TBSYSLB suitable for your installation. A MAINLIB may be allocated with the TLIB parameter, for example, `TBDR TLIB('MY.FAVORITE.MAINLIB')`

The tablesONLINE/ISPF panel has a field to specify a MAINLIB. It allocates CMD, SYSOUT, and TBSYSLB the same as TBDR.

Sample JCL to run TBDRIVER in batch is provided in YOUR.PREFIX.TBASE.CNTL dataset.

At start-up, TBDRIVER opens the TBDRHELP table, issues tableBASE commands ML,MAINLIB and CS, NN (to turn off both the abend-on-error and wait-for-enqueued-table switches) so that command results therefore echo the tableBASE error code. You may change these switch settings, if desired, by the CS command.

Other table libraries may be allocated before invoking TBDR or the ISPF panel, or may be allocated from within TBDRIVER by using the AL command. Other data definitions (TBOPT, TBTSRPT), if desired, must be allocated before invoking TBDRIVER.

TBDRIVER (DK1TDRV) commands

A TBDRIVER (DK1TDRV) command begins with a two-letter command name (in positions 1 and 2 of the input line, see Table 9-2, "TBDRIVER (DK1TDRV) command descriptions," on page 276). If parameters are provided with the command, they are separated from the command and from each other by commas. A null (blank) input line acts as if you had entered a PR command that inherits the previous command's COUNT value. For example, if the previous command was a GL, then the blank line 'PR' does not display any rows, but if the previous command was an open (which sets count to 0), then a blank line PR will display rows starting with Row 1.

TBDRIVER parses the input line into parameter blocks for tableBASE then invokes the tableBASE API (TBLBASE) to carry out the command. Some special TBDRIVER commands, and commands accepting wild-carded table names, perform a sequence of tableBASE calls. The resulting ERROR, FOUND, and COUNT values returned by tableBASE are displayed.

Note: Commands that were unique in VTS in previous releases are now included in the batch TBDRIVER.

In the TBDRIVER command descriptions in Table 9-2, parameters in lower-case are optional; defaults are supplied by tableBASE or a value specified in a previous TBDRIVER command is re-used. For example, in the sequence of commands

```
GD,MYTABLE
OR
CN,,NEWTABLE
```

MYTABLE defaults in the OR and CN commands.

Parameters in upper-case must be specified.

If parameters are omitted, commas must be used as placeholders when other parameters follow in the command, for example, `CG,MYTABLE,,8`.

In all commands except ML, blank spaces may not be used within the command line. Parsing of command options stops at the first blank space.

An asterisk at the beginning of a line may be used to indicate a comment line.

```
* This is a comment.
```

Details on all parameters are given for the corresponding tableBASE commands described in Chapter 3 of this guide.

Table 9-2: TBDRIVER (DK1TDRV) command descriptions

Command name	Format Description
AB [1]	`AB`
	Abend with 0C7 (Used to force a TBDRIVER dump)
AL	`AL,ddname,DSNAME,disp`
	Allocate an existing library • **disp** is S for Shared (default) or O for exclusive
AR[1]	`AR,tablename`
	Open base table and invoke all associated alternate indexes.
BN	`BN`
	Display installation's banner and display tableBASE version
CA	`CA,alttablename,datatable,org,method,keycount,keyloc,keysize`
	Create Alternate Index definition • **keycount** must be set to 1 if **keyloc** or **keysize** is specified
CD	`CD,tablename,org,method,type,smc,rpwsd,wpswd,rowsize,keysize,keyloc,rowestimate,numgens,expandfactor,lowdensity,hidensity,viewversion,viewname,comment,locklatch`
	Change table definition
CG	`CG,tablename,wpswd,gennumber`
	Change number of table generations retained in library
CL [2]	`CL,tablename,locklatch`
	Close table
CN	`CN,tablename,newtablename`
	Change table name in TSR • **CN** changes current tablename to **newtablename**
CS	`CS,aweot`
	Change status switches See "Change Status (CS)" on page 73
DC	`DC,tablename,count,locklatch`
	Delete row by count

Table 9-2: TBDRIVER (DK1TDRV) command descriptions (Continued)

Command name	Format Description
DE	`DE`
	Disengage libraries (Close all library files)
DG	`DG,tablename,wpswd,gennumber`
	Delete a table generation • **gennumber** defaults to zero
DK	`DK,tablename,key,rowlengthoverride,locklatch`
	Delete row by key • **key** defaults to spaces
DL	`DL,ddname`
	Define library
DR [1]	`DR,ddname`
	List next 10 library entries (uses NX and GD) (For interactive, hit <RETURN>; for batch, enter a blank line.)
DT	`DT,tablename,org,method,type,smc,rpswd,wpswd,rowsize,keysize,keyloc,rowestimate,numgens,expandfactor,lowdensity,highdensity,viewversion,viewname,comment,locklatch`
	Define table
DV [2]	`DV,tablename,DDNAME,locklatch`
	Divert table to a library where table does not exist
DW [2]	`DW,tablename,DDNAME,locklatch`
	Divert table to a library where table exists
FC	`FC,tablename,count,rowlengthoverride`
	Fetch row by count

FG	`FG,tablename,KEY,rowlengthoverride,partialkeylength`
	`FG,tablename,PARTIALKEY*,rowlengthoverride`
	`FG,tablename,*,rowlengthoverride`

	Fetch row using generic key • **KEY** and **PARTIALKEY** default to spaces
FK	`FK,tablename,key,rowlengthoverride`
	Fetch row by key • **key** defaults to spaces

Table 9-2: TBDRIVER (DK1TDRV) command descriptions (Continued)

Command name	Format Description
FN	`FN,tablename,key,rowlengthoverride`
	Fetch row or next highest (lowest) by key • **key** defaults to spaces
GD [3]	`GD,tablename,gennumber`
	Get table definition
GF	`GF,tablename,rowlengthoverride`
	Get first table row
GL	`GL,tablename,rowlengthoverride`
	Get last table row
GN	`GN,tablename,rowlengthoverride`
	Get next table row
GP	`GP,tablename,rowlengthoverride`
	Get previous table row

HP [1]	`HP – General Help info`
	`HP,HP – Get list of commands`
	`HP,xx – Get help on command xx`

	Help about command(s)
IA [4]	`IA,alttablename,datatable,org,method,keycount,keyloc,keysize,locklatch`
	Invoke Alternate Index • Keycount must set to 1 if keyloc or keysize specified
IC	`IA, tablename,count,rowdata,rowlengthoverride,locklatch`
	Insert row by count • rowdata defaults to spaces
IK	`IK,tablename,rowdata,rowlengthoverride,locklatch`
	Insert row by key • rowdata defaults to spaces
LD [5]	`LD,tablename,ddname,tablenamemask,dirtype`
	List directory (Create a table from directory contents) • dirtype is V for Views, D for Data tables, T (default) for all generations of all tables

Table 9-2: TBDRIVER (DK1TDRV) command descriptions (Continued)

Command name	Format Description

LG [1]	`LG,tablename,key,rowlengthoverride,partialkeylength`
	`LG,tablename,partialkey*,rowlengthoverride`
	`LG,tablename,*,rowlengthoverride`

	List rows using generic key • **key**, **partialkey** default to spaces
LL	`LL`
	List library search order
LS	`LS`
	List status switches See "Change Status (CS)" on page 73
LT	`LT,howmany,startfrom`
	List tables currently open in the TSR • **howmany** defaults to 10 • **startfrom** defaults to 1
LV	`LV`
	List VTS settings
ML	`ML,LIB1,LIB2,...,LIB10`
	Modify library search order • Up to 10 DDnames and VTS:xxxx • Set empty search list • Blank characters between library names are ignored
MT	`MT,tablename,locklatch`
	Empty table (erase contents in TSR)
NX	`NX,tablename,ddname,s`
	Return next tablename in library • **s** is S for library space stats (tablename is ignored) • **NX** changes the current table name unless S is given
OA[1]	`OA,tablename`
	Open Any. If table is a base table, open table for read. If table is an alternate, open base table and invoke the alternate. Supports initial loading of tables in VTS-TSRs.

Table 9-2: TBDRIVER (DK1TDRV) command descriptions (Continued)

Command name	Format Description
OR [4]	`OR,tablename,password,gennumber,indiropen`
	Open table for read • **indiropen** is I for indirect open
OW [4]	`OW,tablename,wpswd,gennumber,indiropen,locklatch`
	Open table for write • **indiropen** is I for indirect open

PR [1, 2]	`PR,tablename,startrow,endrow,rowlengthoverride`
	`PR,tablename,*,,rowlengthoverride`
	`PR,tablename,startrow,*,rowlengthoverride`

	List next 10 table rows (default command) • **PR** defaults to printing of 10 rows after current row
QR [1]	`QR,tablename,password,gennumber,indiropen`
	Open View for read • **indiropen** is I for indirect open
QW [1]	`QW,tablename,wpswd,gennumber,indiropen,locklatch`
	Open View for write • **indiropen** is I for indirect open
RC	`RC,tablename,count,rowdata,rowlengthoverride,locklatch`
	Replace row by count • **rowdata** defaults to spaces
RF[2]	`RF,tablename`
	Refresh a table that is opened for read in TSR (or VTS-TSR) • This command will cause all opened Alternate indexes to be refreshed as well
RK	`RK,tablename,rowdata,rowlengthoverride,locklatch`
	Replace row by key • **rowdata** defaults to spaces
RL [2]	`RL,tablename,locklatch`
	Release table (Change open status from write to read)

Table 9-2: TBDRIVER (DK1TDRV) command descriptions (Continued)

Command name	Format Description
RN	`RN,tablename,newtablename,wpswd`
	Rename table in library • **RN** changes current tablename to **newtablename**
SI	`SI,criterion`
	Set indirect table search criterion
SK	`SK,tablename,key`
	Search row by Key • **key** defaults to spaces
SP [1]	`SP,1,,,yyyymmdd` (for date-sensitive processing) `SP,0` (for regular processing)
	Set special function processing controls • **yyyymmdd** defaults to current date
SS [1]	`SS,VTSSERVERNAME` `SS` (Unset server name)
	Set VTS name (TB-SUBSYSTEM) in TB-PARM
ST [2]	`ST,tablename,locklatch`
	Store table
UL	`UL,DDNAME`
	Unallocate a library
VE [1]	`VE`
	Show version of DK1TDRV.
VS [1]	`SS,VTSSERVERNAME`
	Synonym for SS [1] - Note that this is a different command from the tableBASE VS command.
VX[1]	`VX,pgmname,text`
	Execution of a program in VTS • **pgmname** is the name of a user-written program • **text** is data to be passed to the program

Table 9-2: TBDRIVER (DK1TDRV) command descriptions (Continued)

Command name	Format Description
XT	`XT,tablename,wpswd`
	Eliminate all generations of a table from library

Note 1: This is a special TBDRIVER (DK1TDRV) command.

Note 2: Asterisks may be used as wildcard characters in the table name. An asterisk (*) at the end of a string denotes zero or more characters. If the asterisk is elsewhere in the string, the asterisk denotes a single character.

Examples:OR,B*Opens all tables beginning with letter B

OR,*S* Opens all tables with S as second letter

OR,*C Opens all tables with 2-character names, C as 2nd

OR,F*T* Opens all tables with F as 1st, T as 3rd character

The input command is processed against all the tables matching the table name mask. For example, when the close table (CL) command is issued to table HOWARD* the command closes all the tables and alternates with HOWARD in the first six characters of the name.

Note 3: The GD command display of the DDNAME field is increased from eight bytes to 12 bytes. This allows display of VTS:xxxxxxxx for tables accessed by a linked VTS-TSR. This may affect applications which post-process the output from TBDRIVER steps.

Note 4: Same as Note 2 but list of tables to be matched is all tables in the current library search list (LIB-LIST).

Note 5: TABLENAMEMASK is a wild-card form as above, or contains no asterisks, for example, CAT in which case it is assumed to be equivalent to CAT*.

The list matched is all tables in the current library search LIB-LIST.

Example using Alternate Indexes

The ability to view a Data Table with a different key or organization or search method using an Alternate Index can be accomplished in one of two ways:

- through a permanent definition (one that is stored on the tableBASE library)
- through a temporary definition (an alternate that has been invoked with a definition supplied at the time of invocation). A permanent Alternate Index may be Opened (OR/OW), or invoked using the IA command; a temporary Alternate Index must be Invoked using IA.

The example uses a sample table called, "EXAMPLE" that is supplied with the product. There are two types of alternate indexes, a permanent alternate index, and a temporary alternate index (see Figure 9-1).

```
OR,EXAMPLE                          - Open Data Table (read)
IA,EXAMALT4,EXAMPLE,S,B,1,1,20      - Temp last name alternate index
IA,EXAMALT5,EXAMPLE,S,B,1,21,14     - Temp first name alternate index
IA,EXAMALT6,EXAMPLE,S,Q,1,35,8      - Temp department alternate index
FG,EXAMALT4,B*                      - Fetch names starting with 'B'
FG,EXAMALT5,JOHN*                   - Fetch first names...(John)
FG,EXAMALT6,ADVTG*                  - Fetch Advertising department
CL,EXAMPLE                          - Remove all accesses
```

Figure 9-1: Temporary Alternate Index

Note: The open mode (OR/OW) of the Data Table or Alternate View is not significant with respect to the examples. However, if you intend to update the changes to the library you must open the Data Table for write using the OW command.

JCL

Figure 9-2 shows an example of a MVS JCL that is used to execute TBDRIVER in batch

```
//STEP1   EXEC PGM=TBDRIVER
//STEPLIB DD DISP=SHR,DSN=*your.prefix*.TBASE.LOAD
//SYSOUT  DD SYSOUT=*
//MAINLIB DD DISP=SHR,DSN=*your.prefix*.TBASE.LIBRARY
//TBSYSLB DD DISP=SHR,DSN=*your.prefix*.TBSYSLB
//TBDUMP  DD SYSOUT=*
//CMD     DD *
TBDRIVER command sequences;
/*
//TBOPT   DD   *                                    (If needed)
overrides for default settings
//TBTSRPT DD SYSOUT=*                               (If needed)
```

Figure 9-2: Sample MVS JCL

10 Error diagnosis

Sometimes tableBASE software errors occur due to user error, environmental situations, or internal errors. A user error may occur when passing incorrect information or the wrong parameter to tableBASE. An unexpected termination of a CICS region, or running out of memory, may lead to environmental errors. These errors may lead to more serious problems for open tables and ongoing transactions. In very rare cases, an internal tableBASE error may occur. These errors should be reported immediately.

tableBASE user errors

Under most circumstances, when tableBASE encounters an error it will return either an error code and error sub-code, or, if the tableBASE settings indicate it should abend rather than continue, it will abend with a user code.

tableBASE user errors may be the result of an incorrectly issued tableBASE command or providing incorrect parameters. These are diagnosed by consulting the resulting error code and error sub-code or the abend user code. These error messages should provide sufficient information to understand and correct the basic user error problems. For a list of tableBASE messages and error codes, see Appendix E on page 407.

Note: Whether or not tableBASE abends when it receives an error can be controlled via the use of a CS command or a temporary Abend Error override. See "Change Status (CS)" on page 73 for information on modifying this Status Switch.

An abend will normally result in a SYSLOG message. The user may provide a JCL file statement with his job in order to obtain the dump that will be produced on SYSUDUMP under MVS and OPTION DUMP.

In addition, provision of a TBDUMP JCL file statement allows the printing of a report containing only the saved registers and tableBASE control blocks (see "TBDUMP Diagnostic Information" on page 287). This will usually be required by DataKinetics Technical Support to diagnose the cause of the error.

Abnormal terminations

tableBASE will attempt to recover from unusual situations that may cause tableBASE errors. However, under certain circumstances (close and open) where tableBASE is unable to recover and it is deemed to be unwise to proceed, tableBASE will terminate regardless of the Status Switch or Abend override settings, and will close the affected table(s).

Under CICS, tableBASE will produce a DKL1 CICS transaction dump for any abnormal termination.

tableBASE internal errors

tableBASE has many internal checks. If an internal tableBASE error does occur it will be identified by an error code above a 0100 (and not 1072). These errors should not normally occur, but if one does occur, please call your tableBASE administrator at once to avoid damage to your data.

Under CICS, tableBASE will produce a CICS transaction dump with an abend code of LGIC when an internal error is trapped.

TBDUMP Diagnostic Information

A logic error is detected when, in DataKinetics's opinion, tableBASE has run into a situation that is not logically consistent with its design. An example might be if a tableBASE TSR indicates it has space available, but when tableBASE scans for empty space, none is found. Normally the task or transaction is abended (G301 or U301) and message DK100301S is issued. In most cases, DataKinetics staff will have to debug these problems because they involve researching the source code modules indicated in the TBDUMP and/or JESMSGLG.

There is information available from a TBDUMP that is not always available from any other dump. The following tableBASE internal control blocks are normally included in a TBDUMP.

Thread Management Area (TMA)

Thread Management Area (TMA)—eyecatcher "TMAAREA". This control block is created on the first call to tableBASE for a thread; it is freed when the thread terminates. In CICS it represents a transaction and is allocated by the tableBASE Task Related User Exit (TRUE) DK1TCRM. In other environments it represents a TCB. In tableBASE modules register 9 usually contains the address of the TMA.

User fields in the TMA:

```
+'0020' copy of the current (or last) tableBASE command issued by the thread
+'0068' copy of the TBPARM (if any)
+'0375' status switches in effect for the thread
+'0382' current library list
```

Note: Important! The displacements above, and in the rest of this section, may change with maintenance to tableBASE. These displacements are valid for Version 6.

Thread Work Area (TWA)

There is no significant user information in this block

Communication Management Area (CMA)

Communication Management Area (CMA)—eyecatcher "CMAAREA". This control block is created when tableBASE is initialized in a region (job step); it is freed at step termination.

User fields in the CMA:

```
+'0020' startup date/time
+'0090' Jobname of application
```



```
+'0098' Stepname of application
+'0208' Level of tableBASE Root module (RTXvrm.ccnnn)
+'0220' Level of tableBASE Nucleus module (NUCvrm.ccnnn)
```

Common Management Extension Area (CME)

Common Management Extension Area (CME). This control block starts at location 0 in the TSR (an MVS dataspace). It contains information about the status of the TSR and has pointers to other areas in the TSR.

Command Trace Area

Command Trace Area—eyecatcher "CMDTRACE". The command trace records the last 10 calls to tableBASE in this region. This area is only present if the TRACE Status Switch (Switch 5) is set to Y. Each entry consists of a before and after image of the command area and TBPARM.

The command trace area has a header which contains an address (@+14) which is used for the next trace entry; so the last entry is the one just before this pointer. The trace table wraps, so there may be a residual trace entry in the "next" entry. If an abend occurs while processing a command, the before entry will be there, but not the after.

```
0000  CMTHEYEC DC    CL8'CMDTRACE'     eyecatcher
0008  CMTHSIZE DS    AL4    Total size of CMDTRACE area
000C  CMTHENTL DS    AL4    Size of each entry (COMMAND + TBPARM)
0010  CMTHSTRT DS    AL4    addr of first trace table entry
0014  CMTHNEXT DS    AL4    addr of next trace table entry to use
0018  start of entries, each of which contains:
      +00      COMMAND AREA  (length 72 bytes)
      +48      TBPARM        (length 64 bytes)
```


11 tableBASE subroutines

Three tableBASE subroutines TBACC, TBINDX and DKTBNAME are detailed in this chapter. In Version 6 these features are unchanged.

Subroutine TBACC

The subroutine TBACC is a low-level access subroutine in tableBASE. It is used independently of tableBASE when a program provides its own storage to maintain table data. TBACC can operate on any memory space which contains repeating groups of fixed length data.

This subroutine can be used to operate on table data that has been loaded into a program area using the DU command of tableBASE. (If used in conjunction with the DU command, see "Dump Table Contents (DU)" on page 86 and "TBACC-DEF (29 bytes)" on page 167).

If you need to manipulate a table through an Index using this low level of access, TBINDX should be used. TBINDX is described later in this chapter.

Summary of TBACC commands

Table 11-1 is a summary of the commands available to the programmer using TBACC. Subsequent sections provide details for their use and explanations of the various table organizations and search methods which can be accommodated.

The commands are described in detail in the following section (see "Command descriptions" on page 290). Field names (e.g., ROW_MA, SUBSCRIPT, etc.), are described in following sections:

- "TBACC parameters" on page 292
- "Calling TBACC" on page 292
- "Table definition parameter description" on page 293.

Table 11-1: TBACC commands

Command		Command description
S	Search	Search for matching row
I	Insert	Insert a row
D	Delete	Delete a row
U	Update	Search and insert (if not found)
R	Remove	Search and delete (if found)
P	Put	Replace in table
T	Take	Retrieve from table
F	Fetch	Search and Take (if found)
A	Arrange	Sort physically
C	Compress	Compress Hash table

Command descriptions

Search (S)

The S command allows a table to be searched for a row matching on the key.

Insert (I)

The I command will cause the row in ROW-WA to be inserted into the table.

An insert into a sequential, hash, or user-controlled sequence table requires SUBSCRIPT to be set by the programmer or by a preceding search. A row inserted into a random table is added to the end of the table, so SUBSCRIPT is ignored. After an insert, check to determine if the insert was successful by checking OVERFLOW-CODE. If the code is set to Y, the insert was not successful because the table was full. A successful insert will cause NO-OF-ROWS to be incremented.

Delete (D)

The D command will delete the row pointed to by SUBSCRIPT. SUBSCRIPT must be set by the programmer or by a preceding search.

A delete from a random table is performed by moving the last row in the table to the row pointed to by SUBSCRIPT. A successful delete will reduce NO-OF-ROWS by one.

Update (U)

The U command comprises a search and an insert. After an update, you must perform two tests:

- Determine whether or not the row was found;
- If the row was not found, determine whether the attempt to insert failed due to an overflow in the size of the table.

If the row was neither found nor forced an overflow, assume that the row has been inserted at the location specified by the value of SUBSCRIPT.

Remove (R)

The R command is a combination of search and delete. The remove was successful if the FOUND-CODE is set to Y. If that code is set to N, the row was not found and could not be deleted.

Put (P)

The P command will cause the row in ROW-WA to be copied to the entry pointed to by SUBSCRIPT. SUBSCRIPT must be set by the programmer or by a preceding search. A successful Put will set the FOUND-CODE to Y. An unsuccessful Put could be caused by trying to replace an empty location in a hash table.

Take (T)

The T command will cause the row pointed to by SUBSCRIPT to be copied to ROW-WA. SUBSCRIPT must be set by the programmer or by a preceding search. If the subscript is zero, low-values will be returned and the FOUND-CODE set to N. Otherwise, the FOUND-CODE will be set to Y.

Fetch (F)

The F command is a combination of search and take. The FOUND-CODE will be set to Y if the Fetch was successful and to N if the row could not be found.

Arrange (A)

The A command will physically sort a table into ascending or descending sequence. Specify organization = S for ascending sequence and organization = D for descending sequence. A hash organized table must be compressed by means of a COMPRESS command before it can be sorted.

Compress (C)

The C command will compress a hash table. After a compression, all empty slots are located at the end of the table. For subsequent use of this table (without sorting), the organization should be set to R and the search method to S.

TBACC parameters

Depending on the command specified, TBACC requires up to four parameters. The parameters required for each command are listed in Table 11-2.

Table 11-2: TBACC parameters

	Command specified									
Parameters	**S**	**I**	**D**	**U**	**R**	**P**	**T**	**F**	**A**	**C**
Table definition	*	*	*	*	*	*	*	*	*	*
Table	*	*	*	*	*	*	*	*	*	*
Search key	*			*	*			*		
Row work area		*		*		*	*	*		

Calling TBACC

TBACC can be called from COBOL and PL/1. The following are examples.

From COBOL

The following is a sample of a TBACC subroutine written in COBOL.

```
CALL 'TBACC'    USING      TABLE-DEFINITION
                           TABLE
                           SEARCH-KEY
                           ROW-WA.
```

From PL/1

The following is a sample of a TBACC subroutine written in PL/1.

```
CALL  TBACC             (TABLE_DEFINITION,
                         TABLE,
                         SEARCH_KEY,
                         ROW_WA);
```


Table definition parameter description

A table definition and a table are required parameters for each call to TBACC. The table definition consists of several fields, some of which you have to specify before calling TBACC, while TBACC will enter information in other fields.

The following fields comprise TBACC-DEF.

1. NO-OF-ROWS (fullword binary)
   Number of valid rows in the table (Maximum 2,147,483,648).

   This field is maintained by TBACC. As rows are inserted by TBACC, it increments this count.

2. ROW-SIZE (fullword binary)
   Specify the number of bytes in each row (Maximum 32,767).

3. KEY-SIZE (fullword binary)
   Specify the number of bytes in the key (Maximum 256).

4. KEY-LOCATION (fullword binary)
   Specify the number of bytes to left of the key in a row.

5. MAX-ROWS (fullword binary)
   Specify the maximum number of rows which can fit into the table (Maximum 2,147,483,648).

6. SUBSCRIPT (fullword binary)
   Set by the programmer before the call or by TBACC as a result of the call.

   After a search, whether by means of an S command or by means of a U, R, or F command, the value of SUBSCRIPT will be set by TBACC.

   If the row was found, SUBSCRIPT will be the subscript of the found row. If more than one row in the table has the required key, the first row in the group will be the one indicated by the value of SUBSCRIPT.

   If the row was not found, SUBSCRIPT will point to the position in the table at which this row may be inserted:

   a. For a randomly organized table, SUBSCRIPT will always equal one more than the number of rows in the table.

   b. For a sequential table, SUBSCRIPT will point to the next higher row in the table. If there is no row, then SUBSCRIPT will equal one more than the number of rows in the table.

c. For a descending sequential table, SUBSCRIPT will point to the next lower row in the table. If there is no row, then SUBSCRIPT will equal one more than the number of rows in the table.

d. For a hash table, SUBSCRIPT will point to an empty space in the table into which the row may be inserted.

e. For a user controlled sequence, SUBSCRIPT will always equal one more than the number of rows in the table.

7. TABLE-ORG (1 byte)
   Specify S, D, H, R, or U.

   Where,

   S = Sequential ascending
   D = Descending sequential
   H = Hash
   R = Random
   U = User controlled

8. FOUND-CODE (1 byte)
   Code returned after Search—Y = found, N = not found
   Code returned after Put—Y = successful, N = failed
   Code returned after Take—Y = successful, N = failed

9. OVERFLOW-CODE (1 byte)
   Code returned after Insert—Y = overflow, N = no overflow

10. ACTION (1 byte)
    Specify the function to be performed.
    S, I, D, U, R, P, T, F, A, or C.

11. SEARCH-METHOD (1 byte)
    Specify S, Q, B, C, or H.

    Where,

    S = Sequential
    Q = Queued Sequential
    B = Binary
    C = Address Tree Binary
    H = Hash

12. WORK-FIELD (3 bytes) Default = LOW-VALUES

This is used by TBACC.

13. WORK-AREA(n bytes) Default = LOW-VALUES
    Required only when METHOD= C or H
    If METHOD = H, n = 12
    If METHOD = C, n = 132.

    (See Warning 5 below.)

Warnings and restrictions

The following warnings and/or restrictions apply to TBACC:

1. Errors in the specification of the parameters will produce indeterminate results.

2. When organization = H and action is I or U then TBACC will return an overflow condition when the number of rows in a table is one less than the maximum. This method of processing maintains the integrity of the search mechanism.

3. When organization = H the table must be initialized to LOW-VALUES (binary zeroes).

4. When organization = H no search key may equal LOW-VALUES (binary zeroes).

5. If for any reason it is necessary to change ROW-SIZE or MAX- ROWS during program execution, sub-field 13, WORK-AREA, in the table definition (TBACC-DEF) must be re-initialized to LOW-VALUES (binary zeroes). Except as noted above this field must never be modified.

Sample COBOL program code using TBACC

The following is a sample of a COBOL program using TBACC commands:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SAMPLE.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

  01  NOTE1                          PIC X(32) VALUE
                                     'SAMPLEWORKINGSTORAGESTARTS'.
  01  xxxx-TBL-DEF.
      05 xxxx-ROWS                   PIC S9(8)COMPVALUE+0.
      05 xxxx-SIZE                   PIC S9(8)COMPVALUE+100.
      05 xxxx-KEYSIZE                PIC S9(8)COMPVALUE+7.
      05 xxxx-KEYLOC                 PIC S9(8)COMPVALUE+4.
      05 xxxx-MAX                    PIC S9(8)COMPVALUE+1000.
      05 xxxx-SUB                    PIC S9(8)COMPVALUE+0.
      05 xxxx-ORG                    PIC XVALUE'S'.
      05 xxxx-FOUND                  PIC X.
      05 xxxx-OVFLOW                 PIC X.
      05 xxxx-ACTION                 PIC XVALUE'S'.
      05 xxxx-METHOD                 PIC XVALUE'C'.
      05 RESERVED                    PIC X(111)VALUELOW-VALUES.
  01  TBL-AREA.
      05 TBL-X                       PIC X(100) OCCURS 1000.
  01  ROW-AREA.
      05 FILLER                      PIC xxxx.
      05 ROW-KEY                     PIC X(7).
      05 FILLER                      PIC X(89).
  01  NOTE2                          PIC X(32)VALUE
                                     'SAMPLEWORKINGSTORAGEENDS'.

 PROCEDURE DIVISION.
*************************************************************
* NORMAL PROCESSING IN WHICH TABLE IS BUILT OR OBTAINED
*************************************************************

*************************************************************
* SEARCH (S)
*************************************************************
    MOVE 'S'                         TO xxxx-ACTION.
    CALL 'TBACC' USING               xxxx-TBL-DEF
                                     TBL-AREA
                                     ROW-KEY.
    IF  xxxx-FOUND = 'Y'
        PERFORM FOUND-ROUTINE
    ELSE
        PERFORM NOT-FOUND-ROUTINE.
```



```
*************************************************************
* INSERT (I)
*************************************************************

    MOVE 'I'                       TO xxxx-ACTION.
    CALL 'TBACC' USING             xxxx-TBL-DEF
                                   TBL-AREA
                                   ROW-AREA.
    IF xxxx-OVFLOW = 'Y'
       PERFORM OVER-FLOW-ROUTINE.

*************************************************************
* DELETE (D)
*************************************************************

    MOVE 'D'                       TO xxxx-ACTION.
    CALL 'TBACC' USING             xxxx-TBL-DEF
                                   TBL-AREA.

*************************************************************
* SEARCH & INSERT (U)
*************************************************************

    MOVE 'U'                          TO xxxx-ACTION.
    CALL 'TBACC' USING                xxxx-TBL-DEF
                                      TBL-AREA
                                      ROW-KEY
                                      ROW-AREA.
    IF  xxxx-FOUND = 'Y'
        PERFORM FOUND-ROUTINE.

*
* IF NOT FOUND CHECK WHETHER TABLE OVERFLOWED.
*

    IF xxxx-OVFLOW = 'Y'
       PERFORM OVER-FLOW-ROUTINE
    ELSE
       PERFORM NOT-FOUND-ROUTINE.

*************************************************************
* SEARCH & DELETE (R)
*************************************************************

    MOVE 'R'                          TO xxxx-ACTION.
    CALL 'TBACC' USING                xxxx-TBL-DEF
                                      TBL-AREA
                                      ROW-KEY.
    IF xxxx-FOUND = 'Y'
       PERFORM FOUND-ROUTINE
    ELSE
       PERFORM NOT-FOUND-ROUTINE.
```

```
************************************************************
* TAKE (T) --- COPY FROM TBL-X (xxxx-SUB) TO ROW-AREA
************************************************************
   MOVE 'T'                          TO xxxx-ACTION.
   CALL 'TBACC' USING                xxxx-TBL-DEF
                                     TBL-AREA
                                     ROW-AREA.

************************************************************
* FETCH (F) --- SEARCH FOLLOWED BY A TAKE IF FOUND
************************************************************

   MOVE 'F'                          TO xxxx-ACTION.
   CALL 'TBACC' USING                xxxx-TBL-DEF
                                     TBL-AREA
                                     ROW-KEY
                                     ROW-AREA.
   IF xxxx-FOUND = 'Y'
      PERFORM FOUND-ROUTINE-1
   ELSE
      PERFORM NOT-FOUND-ROUTINE.

************************************************************
* PUT (P) --- COPY FROM ROW-AREA TO TBL-X (xxxx-SUB)
************************************************************

   MOVE 'P'                          TO xxxx-ACTION.
   CALL 'TBACC' USING                xxxx-TBL-DEF
                                     TBL-AREA
                                     ROW-AREA.

************************************************************
* ARRANGE (A)
************************************************************

   MOVE 'A'                          TO xxxx-ACTION.
   CALL 'TBACC' USING                xxxx-TBL-DEF
                                     TBL-AREA.
```



```
*************************************************************
* COMPRESS (C)
*************************************************************

    MOVE 'C'                          TO xxxx-ACTION.
    CALL 'TBACC' USING                xxxx-TBL-DEF
                                      TBL-AREA.
 FOUND-ROUTINE.

*    USING THIS FOUND ROUTINE, THE FOUND ROW MUST BE
*    ADRESSED IN THE TABLE BY MEANS OF SUBSCRIPTING,
*    OR MOVED OUT INTO A WORK AREA.

 FOUND-ROUTINE-1.

*    USING THIS FOUND ROUTINE, THE FOUND ROW IS IN
*    ROW-AREA.

 NOT-FOUND-ROUTINE.

 OVERFLOW-ROUTINE.

*
*************************************************************
```

Subroutine TBINDX

The subroutine TBINDX is an access subroutine that can be used when a program provides its own storage to maintain table data. TBINDX can operate on any area that contains repeating groups of fixed length data.

When many table insertions are required, or when frequent reorganization is required, TBINDX allows more efficient processing of tables than TBACC. TBINDX also reduces the storage needed for hash-organized tables.

As with TBACC, TBINDX can be used to operate on table data that has been loaded into a program area using the DU command (see "Dump Table Contents (DU)" on page 86).

The subroutine TBINDX uses a separate table to store the Index(es). This table is referred to as a tag table. The table data that you wish to access is stored in a Data Table (also known as a primary or base table). The tag table contains an Index into the primary table. For each entry in the primary table there is an associated eight byte entry in the tag table. The first four bytes of this entry contain the address of the key of the associated primary table entry relative to the beginning of the primary table. Bytes five through eight of the tag table entry contain the full word binary subscript of the associated primary table entry.

Summary of TBINDX commands

Table 11-3 is a summary of the commands available to the programmer using TBINDX. Subsequent sections provide details for command use and explanations of the various table organizations and search methods which can be accommodated.

Table 11-3: TBINDX commands

Command		Command description
S	Search	Search for matching row
I	Insert	Insert a row
D	Delete Direct	Delete a row directly
E	Delete Indirect	Delete a row indirectly
U	Update	Search and insert (if not found)
R	Remove	Search and delete (if found)
P	Put Direct	Replace in table
Q	Put Indirect	Replace in table
T	Take Direct	Retrieve from table
V	Take Indirect	Retrieve from table
F	Fetch	Search and take (if found)
G	Generate tag table	Create an index (tag) table
H	Hash in place	ORG must = 'H')
A	Arrange	Sort physically

The commands are described in detail in the following section (see "Command descriptions" on page 301). Field names (e.g., ROW_WA, SUBSCRIPT, etc., are described in following sections:

- "TBINDX parameters" on page 304
- "Calling TBINDX from COBOL" on page 304
- "Table definition parameter description" on page 304

Command descriptions

Search (S)

The Search (S) command will cause the tag table to be searched for a match on the key.

Insert (I)

The Insert (I) will cause the row in ROW-WA to be inserted into the first empty primary table location—normally this is at the end. The Index to that primary table entry is placed in the tag table at the location pointed to by TAG-SUBSCRIPT.

If multiple tag tables are being used to access a table (reflecting multiple Indexes), each tag table must reflect the fact that a row has been inserted into the primary table. In this case a call to TBINDX is required using the same primary table and ROW-WA, but with each different tag table. To achieve this, subtract 1 from the NO-OF-ROWS before each of the subsequent calls to TBINDX using a different tag table. Since the inserted row is always placed at the end of the Data Table (primary table), the subsequent calls will not cause duplicate entries.

An insert into a sequential or hash table requires TAG-SUBSCRIPT to be set by the programmer or by a preceding search. When inserting into a random table the Index to the primary table entry is added to the end of the tag table, so TAG-SUBSCRIPT is ignored.

After an insert, check to determine if the insert was successful by checking OVERFLOW-CODE. If the code is set to Y, the insert was not successful because the primary or tag table was full.

A successful insert will cause NO-OF-ROWS to be incremented.

Delete (D)

The Delete (D) command will delete the row in the primary table pointed to by PRIMARY-SUBSCRIPT. PRIMARY-SUBSCRIPT must be set by the programmer or by a preceding search. The Index in the tag table (which points to this entry) will also be deleted. If the row is a hash table empty slot, the FOUND-CODE will be set to 'N' and the delete will not be done.

Delete will reduce NO-OF-ROWS by one.

Delete Indirect (E)

The Delete Indirect (E) command will delete the row in the primary table pointed to by the Tag Table entry pointed to by TAG-SUBSCRIPT. TAG-SUBSCRIPT must be set by the programmer or by a preceding search. The Index in the tag table pointed to by TAG-SUBSCRIPT will also be deleted.

Delete Indirect will reduce NO-OF-ROWS by one.

Update (U)

The Update (U) command (if not found) is a combination of search and insert. After an update, two tests must be done:

- Determine whether or not the row was found by testing the FOUND-CODE;
- If the row was not found, determine whether the attempt to insert failed due to an overflow in the size of the primary or tag table by testing the OVERFLOW-CODE.

Remove (R)

The Remove (R) command is a combination of search and delete.

The action was successful if the FOUND-CODE is set to Y. If that code is set to N, the row was not found and could not be deleted.

Put Direct (P)

The Put Direct (P) command will cause the row in ROW-WA to be moved to the primary table entry pointed to by PRIMARY-SUBSCRIPT. PRIMARY-SUBSCRIPT must be set by the programmer or by a preceding search.

Put Indirect (Q)

The Put Indirect (Q) command will cause the row in ROW-WA to be moved to the primary table entry pointed to by the Tag Table entry pointed to by TAG-SUBSCRIPT. TAG-SUBSCRIPT must be set by the programmer or by a preceding search. The TAG-SUBSCRIPT must not point to an empty hash row or the FOUND-CODE will be set to N and the Put will not be done.

Take Direct (T)

The Take Direct (T) command will cause the primary table entry pointed to by PRIMARY- SUBSCRIPT to be moved to ROW-WA. PRIMARY-SUBSCRIPT must be set by the programmer or by a preceding search.

Take Indirect (V)

The Take Indirect (V) command will cause the primary table entry pointed to by the Tag Table entry pointed to by TAG-SUBSCRIPT to be moved to ROW-WA. TAG-SUBSCRIPT must be set by the programmer or by a preceding search.

Fetch (F)

The Fetch (F) command is a combination of Search and Take. The FOUND-CODE will be set to Y if the Fetch was successful and to N if the row could not be found.

Generate Tag Table (G)

The Generate Tag Table (G) command will initialize a tag table appropriate to the organization of the table.

Unless the primary table is empty, the tag table must be initialized before any searching or inserting is done. Normally this tag table generation need only be done once for each tag table, at the beginning of the program.

Regenerate the tag table if any of the following fields change in value during the course of the program execution:

- ROW-SIZE
- KEY-SIZE
- KEY-LOCATION
- TABLE-ORG
- TAG-TBL-MAX (if ORG = H).

For a sequential table, a G command will result in a table in which the first entry in the tag table "points to" (by means of the address and subscript described above) the lowest entry in the primary table; the second entry in the tag table points to the second lowest entry in the primary table; the last entry in the tag table points to the highest entry in the primary table.

For a non-sequential table, no reliance should be placed on the order of rows. All subscripts of zero should be ignored.

Hash In Place (H)

The Hash In Place (H) command will convert a sequential or random table into a hash table. An organization and method of H must be specified. A work area equal in size to one row is required. The area used for the insert row, ROW-WA, will suffice.

Hash in Place is only valid for Data Tables (primary tables), not for tag tables.

Arrange (A)

The Arrange (A) command will physically sort a primary table into ascending or descending sequence. Specify organization = S for ascending sequence and organization = D for descending sequence.

TBINDX parameters

Depending on the command specified, TBINDX requires up to five parameters. The parameters required for each command are listed in Table 11-4:

Table 11-4: TBINDEX parameters

	Command specified													
Parameters	**S**	**I**	**D**	**E**	**U**	**R**	**P**	**Q**	**T**	**V**	**F**	**G**	**H**	**A**
Table definition	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Table	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Tag table	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Search key	*				*	*					*			
Row work area		*			*		*	*	*	*	*		*	*

Calling TBINDX from COBOL

The following is a sample of a TBINDX subroutine written in COBOL.

```
CALL 'TBINDX' USING       TABLE-DEFN
                          TABLE
                          TAG-TABLE
                          SEARCH-KEY
                          ROW-WA.
```

Table definition parameter description

In the example above, a table definition and the name of the primary and tag tables are required parameters for each call to TBINDX. The table definition consists of several fields, some of which you have to specialize before calling TBINDX, while TBINDX will enter information in other fields.

The following fields comprise TBINDX-DEF:

1. NO-OF-ROWS (fullword binary)

   Number of valid rows in the primary table (Maximum 2,147,483,648)

   This field is maintained by TBINDX. As rows are inserted by TBINDX, it increments this count.

2. ROW-SIZE (fullword binary)
   Specify the number of bytes in each row of the primary table

   (Maximum 32,767).

3. KEY-SIZE (fullword binary)
   Specify the number of bytes in the key (Maximum 256).

4. KEY-LOCATION (fullword binary)
   Specify the number of bytes to left of the key in a row.

5. MAX-ROWS (fullword binary)
   Specify the maximum number of rows which can fit into the primary table (Maximum 2,147,483,648).

6. TAG-SUBSCRIPT (fullword binary)
   This is the current position in the tag table. It is set by the programmer before the call or by TBINDX as a result of a call.

   After a search, whether by means of an S, U, or F command, TAG-SUBSCRIPT will be set by TBINDX. If the search resulted in a found condition, TAG-SUBSCRIPT will be equal to the subscript in the tag table of the subscript of the row in the primary table. If the row was not found, TAG-SUBSCRIPT will point to the location in the tag table where the Index to the row in the primary table should be inserted.

   In addition, if the row is in the table, (as a result of a successful search or a successful insert) PRIMARY-SUBSCRIPT will be equal to the subscript of the found row in the primary table.

7. TABLE-ORG (1 byte)
   Specify S, D, H, R, or U

   Where,

   S = Sequential ascending
   D = Descending sequential
   H = Hash
   R = Random
   U = User controlled

8. FOUND-CODE (1 byte)
   Code returned after Search. Y = found, N = not found
   Code returned after Put direct. Y = successful, N = failed

9. OVERFLOW-CODE (1 byte)
   Returned after Insert. Y = overflow, N = no overflow

10. ACTION (1 byte)
    Function to be performed.
    S, I, D, E, U, R, P, Q, T, V, F, G, H, or A.

11. SEARCH METHOD (1 byte)
    Specify search method: S, Q, B, C, or H.

    Where,

    S = Sequential
    Q = Queued Sequential
    B = Binary
    C = Address Tree Binary
    H = Hash

12. WORK-FIELD (3 bytes) Default=LOW-VALUES

13. PRIMARY-SUBSCRIPT (fullword binary)
    This is the position in the Data (or primary) table. Set by the programmer before the call or by TBINDX as a result of the call.

A work area of 220 bytes is required by TBINDX. It must be defined as either Field 14 or as Fields 15a-c, as shown below.

Either Field 14:

14. WORK-AREA (220 bytes) Default=LOW-VALUES
    Required except when TABLE-ORG = 'H'

    (See "Warnings and restrictions" on page 307.)

Or one of Fields 15a, 15b & 15c (areas required only when TABLE-ORG = 'H'):

15.a) TAG-TABLE-MAX (fullword binary)
    Maximum number of rows which can fit into the Tag Table

15.b) WORK-AREA-1 (fullword binary) VALUE +0
    This field must be reset to 0 if any changes occur to TAG-TABLE-MAX or TABLE-ORG.

15.c) WORK-AREA-2 (212 bytes.)

Warnings and restrictions

The following warnings and/or restrictions apply to TBINDX:

- Errors in the specification of the parameters will produce indeterminate results.
- When organization = 'H' and action is 'I' or 'U', TBINDX will return an overflow condition when the number of rows in the tag table is one less than the tag table maximum. This is in order to maintain the integrity of the search mechanism.
- The overflow condition will be set if the primary table becomes full.
- When organization = 'H', no search key may equal LOW-VALUES (binary zeroes).
- TAG-TABLE-MAX, when specified must be greater than MAX-ROWS.
- If for any reason it is necessary to change ROW-SIZE or MAX- ROWS (for a table with METHOD = 'C') during program execution, field 14 in the definition must be re-initialized to LOW-VALUES (binary zeroes). Except as noted above this field must never be modified.
- Within the table definition, either Field 14 or Fields 15a to 15c are required, not both.
- If an unrecoverable inconsistency is discovered between the tag and primary tables, Register 15 will be set to 4 to indicate an error. This value can be checked from COBOL using the RETURN-CODE field.

Sample COBOL program using TBINDX

The following is a sample of a COBOL program using TBINDX commands.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SAMPLE.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 P-4                          PIC X(32) VALUE 'SAMPLE WORKING STORAGE STARTS'.

*************************************************************
*  FOR KEY 1
*************************************************************

 01 X1-TBL-DEF.
    05  X1-N                    PIC S9(9) COMP VALUE +0.
    05  X1-RSZ                  PIC S9(9) COMP VALUE +100.
    05  X1-ASZ                  PIC S9(9) COMP VALUE +7.
    05  X1-ALOC                 PIC S9(9) COMP VALUE +4.
    05  X1-M                    PIC S9(9) COMP VALUE +1000.
    05  X1-TAG-I                PIC S9(9) COMP VALUE +0.
    05  X1-ORG                  PIC X VALUE 'S'.
    05  X1-FND-CD               PIC X.
```

```
        88  X1-FND               VALUE 'Y'.
    05  X1-OVFL-CD               PIC X.
        88  X1-OVFL              VALUE 'Y'.
    05  X1-ACTION                PIC X VALUE 'S'.
    05  X1-METHOD                PIC X VALUE 'C'.
    05  FILLER                   PIC XXX VALUE LOW-VALUES.
    05  X1-PRIMARY-I             PIC S9(9) COMP VALUE +0.
    05  FILLER                   PIC X(220) VALUE LOW-VALUES.
 01 TAG-AREA-1.
    05  FILLER                   PIC X(8) OCCURS 1000.

*************************************************************
*  FOR KEY 2
*************************************************************
 01 X2-TBL-DEF.
    05  X2-N                         PIC S9(9) COMP VALUE +0.
    05  X2-RSZ                       PIC S9(9) COMP VALUE +100.
    05  X2-ASZ                       PIC S9(9) COMP VALUE +6.
    05  X2-ALOC                      PIC S9(9) COMP VALUE +16.
    05  X2-M                         PIC S9(9) COMP VALUE +1000.
    05  X2-TAG-I                     PIC S9(9) COMP VALUE +0.
    05  X2-ORG                       PIC X VALUE 'S'.
    05  X2-FND-CD                    PIC X.
        88  X2-FND                   VALUE 'Y'.
    05  X2-OVFL-CD                   PIC X.
        88  X2-OVFL                  VALUE 'Y'.
    05  X2-ACTION                    PIC X VALUE 'S'.
    05  X2-METHOD                    PIC X VALUE 'C'.
    05  FILLER                       PIC XXX VALUE LOW-VALUES.
    05  X2-PRIMARY-I                 PIC S9(9) COMP VALUE +0.
    05  FILLER                       PIC X(220) VALUE LOW-VALUES.
 01 TAG-AREA-2.
    05  FILLER                       PIC X(8) OCCURS 1000.

*************************************************************
* FOR KEY 3
*************************************************************

 01 X3-TBL-DEF.
    05  X3-N                         PIC S9(9) COMP VALUE +0.
    05  X3-RSZ                       PIC S9(9) COMP VALUE +100.
    05  X3-ASZ                       PIC S9(9) COMP VALUE +10.
    05  X3-ALOC                      PIC S9(9) COMP VALUE +36.
    05  X3-M                         PIC S9(9) COMP VALUE +1000.
    05  X3-TAG-I                     PIC S9(9) COMP VALUE +0.
    05  X3-ORG                       PIC X VALUE 'S'.
    05  X3-FND-CD                    PIC X.
        88  X3-FND                   VALUE 'Y'.
    05  X3-OVFL-CD                   PIC X.
        88  X3-OVFL                  VALUE 'Y'.
    05  X3-ACTION                    PIC X VALUE 'S'.
    05  X3-METHOD                    PIC X VALUE 'C'.
    05  FILLER                       PIC XXX VALUE LOW-VALUES.
    05  X3-PRIMARY-I                 PIC S9(9) COMP VALUE +0.
    05  FILLER                       PIC X(220) VALUE LOW-VALUES.
01  TAG-AREA-3.
    05  FILLER                       PIC X(8) OCCURS 1000.
```



```
*************************************************************
* FOR KEY 1 BUT FOR HASH ORGANIZATION
*************************************************************

 01 XH-TBL-DEF.
    05  XH-N                        PIC S9(9) COMP VALUE +0.
    05  XH-RSZ                      PIC S9(9) COMP VALUE +100.
    05  XH-ASZ                      PIC S9(9) COMP VALUE +7.
    05  XH-ALOC                     PIC S9(9) COMP VALUE +4.
    05  XH-M                        PIC S9(9) COMP VALUE +1000.
    05  XH-TAG-I                    PIC S9(9) COMP VALUE +0.
    05  XH-ORG                      PIC X VALUE 'H'.
    05  XH-FND-CD                   PIC X.
        88 XH-FND                   VALUE 'Y'.
    05  XH-OVFL-CD                  PIC X.
        88 XH-OVFL                  VALUE 'Y'.
    05  XH-ACTION                   PIC X VALUE 'S'.
    05  XH-METHOD                   PIC X VALUE 'H'.
    05  FILLER                      PIC XXX VALUE LOW-VALUES.
    05  XH-PRIMARY-I                PIC S9(9) COMP VALUE +0.
    05  XH-TAG-MAX                  PIC S9(9) COMP VALUE +1500.
    05  FILLER                      PIC X(4) VALUE LOW-VALUES.
    05  FILLER                      PIC X(212) VALUE LOW-VALUES.
 01 TAG-AREA-H.
    05  FILLER                      PIC X(8) OCCURS 1500.
 01 TBL-AREA.
    05  TBL-X                       PIC X(100) OCCURS 1000.
 01 ROW-AREA.
    05  FILLER                      PIC X(4).
    05  ROW-KEY-1                   PIC X(7).
    05  FILLER                      PIC X(5).
    05  ROW-KEY-2                   PIC X(6).
    05  FILLER                      PIC X(14).
    05  ROW-KEY-3                   PIC X(10).
    05  FILLER                      PIC X(54).

 PROCEDURE DIVISION.

*************************************************************
* NORMAL PROCESSING IN WHICH TABLE IS BUILT OR OBTAINED.
*************************************************************

*************************************************************
*    INITIALIZE TAG TABLES
*************************************************************
  MOVE 'G'                           TO X1-ACTION.
  CALL 'TBINDX' USING                X1-TBL-DEF
                                     TBL-AREA
                                     TAG-AREA-1.

  MOVE 'G'                           TO X2-ACTION.
  CALL 'TBINDX' USING                X2-TBL-DEF
                                     TBL-AREA
                                     TAG-AREA-2.

  MOVE 'G'                           TO X3-ACTION.
```

```
  CALL 'TBINDX' USING                X3-TBL-DEF
                                     TBL-AREA
                                     TAG-AREA-3.
*************************************************************
*    SEARCH USING KEY-1
*************************************************************

    MOVE 'S'                         TO X1-ACTION.
    CALL 'TBINDX' USING              X1-TBL-DEF
                                     TBL-AREA
                                     TAG-AREA-1
                                     ROW-KEY-1.
    IF  X1-FND
        PERFORM FOUND-ROUTINE
    ELSE
        PERFORM NOT-FOUND-ROUTINE.

*************************************************************
*    SEARCH USING KEY-2
*************************************************************

    MOVE 'S'                         TO X2-ACTION.
    CALL 'TBINDX' USING              X2-TBL-DEF
                                     TBL-AREA
                                     TAG-AREA-2
                                     ROW-KEY-2.
    IF  X2-FND
        PERFORM FOUND-ROUTINE
    ELSE
        PERFORM NOT-FOUND-ROUTINE.

*************************************************************
*    SEARCH USING KEY-3
*************************************************************
    MOVE 'S'                       TO X3-ACTION.
    CALL 'TBINDX' USING            X3-TBL-DEF
                                   TBL-AREA
                                   TAG-AREA-3
                                   ROW-KEY-3.
    IF  X3-FND
        PERFORM FOUND-ROUTINE
    ELSE
        PERFORM NOT-FOUND-ROUTINE.

*************************************************************
*    SEARCH USING KEY-1  BUT WITH HASH TABLE ORGANIZATION
*************************************************************

    MOVE 'S'                       TO XH-ACTION.
    CALL 'TBINDX' USING            XH-TBL-DEF
                                   TBL-AREA
                                   TAG-AREA-H
                                   ROW-KEY-1.
    IF  XH-FND
        PERFORM FOUND-ROUTINE
    ELSE
        PERFORM NOT-FOUND-ROUTINE.
```



```
*************************************************************
*    INSERT USING LOCATION FOUND BY MEANS OF PREVIOUS
*    SEARCH USING KEY-1.
*************************************************************

   MOVE 'I'                       TO X1-ACTION.
   CALL 'TBINDX' USING            X1-TBL-DEF
                                  TBL-AREA
                                  TAG-AREA-1
                                  ROW-AREA.

   IF X1-OVFL
      PERFORM OVER-FLOW-ROUTINE.


*************************************************************
*    INSERT USING LOCATION FOUND BY MEANS OF PREVIOUS SEARCH
*    USING KEY-2.
*************************************************************

   MOVE 'I'                       TO X2-ACTION.
   CALL 'TBINDX' USING            X2-TBL-DEF
                                  TBL-AREA
                                  TAG-AREA-2
                                  ROW-AREA.

    IF   X2-OVFL
         PERFORM OVER-FLOW-ROUTINE.


*************************************************************
*    INSERT USING LOCATION FOUND BY MEANS OF PREVIOUS
*    SEARCH USING KEY-3.
*************************************************************

   MOVE 'I' TO                        X3-ACTION.
   CALL 'TBINDX' USING                X3-TBL-DEF
                                      TBL-AREA
                                      TAG-AREA-3
                                      ROW-AREA.

   IF  X3-OVFL
      PERFORM OVER-FLOW-ROUTINE.

*************************************************************
*    INSERT USING LOCATION FOUND BY MEANS OF PREVIOUS
*    UNSUCCESSFUL SEARCH USING KEY-1.
*************************************************************

   MOVE 'I'                           TO XH-ACTION.
   CALL 'TBINDX' USING                XH-TBL-DEF
                                      TBL-AREA
                                      TAG-AREA-H
                                      ROW-AREA.
   IF   XH-OVFL
        PERFORM OVERFLOW-ROUTINE.
```

```
*************************************************************
*    SEARCH & INSERT (U) USING KEY-1
*************************************************************

    MOVE 'U'                          TO X1-ACTION.
    CALL 'TBINDX' USING               X1-TBL-DEF
                                      TBL-AREA
                                      TAG-AREA-1
                                      ROW-KEY-1
                                      ROW-AREA.

    IF  NOT X1-FND
        PERFORM NOT-FOUND-ROUTINE.

*
* IF FOUND CHECK WHETHER TABLE OVERFLOWED.
*

    IF  X1-OVFL
        PERFORM OVER-FLOW-ROUTINE
    ELSE
        PERFORM FOUND-ROUTINE.


*************************************************************
*    SEARCH & INSERT (U) USING KEY-2
*************************************************************

    MOVE 'U'                        TO X2-ACTION.
    CALL 'TBINDX' USING             X2-TBL-DEF
                                    TBL-AREA
                                    TAG-AREA-2
                                    ROW-KEY-2
                                    ROW-AREA.

    IF  NOT X2-FND
        PERFORM NOT-FOUND-ROUTINE.

*
*  IF  FOUND CHECK WHETHER TABLE OVERFLOWED.
*

    IF  X2-OVFL
        PERFORM OVER-FLOW-ROUTINE
    ELSE
        PERFORM FOUND-ROUTINE.


*************************************************************
*    SEARCH & INSERT (U) USING KEY-3
*************************************************************

    MOVE 'U'                        TO X3-ACTION.
    CALL 'TBINDX' USING             X3-TBL-DEF
                                    TBL-AREA
```



```
                                    TAG-AREA-3
                                    ROW-KEY-3
                                    ROW-AREA.

     IF  NOT X3-FND
         PERFORM NOT-FOUND-ROUTINE.

    *
    *    IF FOUND, CHECK WHETHER TABLE OVERFLOWED.
    *

     IF  X3-OVFL
         PERFORM OVER-FLOW-ROUTINE
     ELSE
         PERFORM FOUND-ROUTINE.


*****************************************************************
*   SEARCH & INSERT (U) USING KEY-1 BUT WITH HASH ORGANIZATION
*****************************************************************

     MOVE 'U'                       TO XH-ACTION.

     CALL 'TBINDX' USING            XH-TBL-DEF
                                    TBL-AREA
                                    TAG-AREA-H
                                    ROW-KEY-1
                                    ROW AREA.

     IF  NOT XH-FND
         PERFORM NOT-FOUND-ROUTINE.

    *
    *    IF FOUND CHECK WHETHER TABLE OVERFLOWED.
    *

     IF  XH-OVFL
         PERFORM OVER-FLOW-ROUTINE
     ELSE
         PERFORM FOUND-ROUTINE.

*****************************************************************
*    GENERATE INDICES
*IF ORG = R INDICES ARE GENERATED IN SAME SEQUENCE
*AS PRIMARY TABLE.
*IF ORG = S INDICES WILL BE SORTED.
*IF ORG = H A HASH INDEX TABLE IS GENERATED.
*****************************************************************

     MOVE 'G'                         TO XH-ACTION.
     CALL 'TBINDX' USING              X1-TBL-DEF
                                      TBL-AREA
                                      TAG-AREA-1.
```

```
************************************************************
*    HASH IN PLACE
************************************************************

     MOVE 'H'                          TO XH-ACTION.
     CALL 'TBINDX' USING               XH-TBL-DEF
                                       TBL-AREA
                                       TAG-AREA-H
                                       ROW-AREA.

     FOUND-ROUTINE.
     NOT-FOUND-ROUTINE.
     OVER-FLOW-ROUTINE.
```

Subroutine DKTBNAME

The subroutine DKTBNAME provides table name conversion. When writing special purpose programs for tablesONLINE/CICS (an optional tableBASE component) user exits, it may be necessary to generate special table names either to access the internal tables of tablesONLINE/CICS, or to generate temporary tables or Alternate Index tables that follow the naming convention provided by this program. The program DKTBNAME can be used for this purpose.

The generated names are derivatives of the original names. DKTBNAME accomplishes this by changing the bit settings of the first two bits of an 8-byte name.

The program DKTBNAME can be used to query, manipulate, or flip the bit settings of the first two bits of an 8-byte name.

Parameter description

The program takes two parameters: TABLE-NAME and STATUS-BYTE.

- TABLE-NAME is a string of eight bytes that specifies the name of the data or special table.
- STATUS-BYTE (optional) is a one byte indicator that determines the mode of operation.

Three operating modes

The subroutine DKTBNAME can be in one of three operating modes. If the STATUS-BYTE parameter is present, it is either in Query or Manipulate mode. If the STATUS-BYTE parameter is omitted, it is in Flip mode.

Query mode

Query mode is used to discover the type of name you have – Data Table, View, Alternate Index or Alternate Index View. You specify Query mode by entering a question mark (?) in the STATUS-BYTE.

Query mode will return one of the following one byte codes in the STATUS-BYTE (Table 11-5). The content of the first parameter, TABLE-NAME, is NOT modified.

Table 11-5: Query mode

STATUS-BYTE (input)	STATUS-BYTE (output)	First 2 bits of TABLE-NAME (table name is not modified)
?	Blank	On-On (Data Table name)
	F	On-Off (View name)

Note: "On" = bit set to 1, and "Off" = bit set to 0.

Manipulate mode

The STATUS-BYTE will determine the action to be performed on the first two bits of the TABLE-NAME (First parameter). Table 11-6 shows the values of STATUS-BYTE and the result for Manipulate mode.

Table 11-6: Manipulate mode

STATUS-BYTE (input)	Setting of first 2 bits of TABLE-NAME (after manipulation)
blank	On-On (Data Table name)
F	On-Off (View name)
A	Off-On (Alternate Index name)
X	Off-Off (Alternate Index View name)
1	First bit on, second bit as is.
2	Second bit on, first bit as is.

Flip mode

When the STATUS-BYTE parameter is omitted, the second bit of the table name is flipped from On to Off; or from Off to On. In essence, the table name is flipped from a Data Table to a View, or vice versa.

COBOL examples using DKTBNAME

The following example changes a Data Table name to a View name by omitting the STATUS-BYTE parameter from the call, thus moving into FLIP mode.

```
   05 WS-TABLE-NAME                 PIC X(8).
   05 WS-STATUS-BYTE                PIC X.

etc.

   MOVE 'TABLE1' TO                 WS-TABLE-NAME.
   CALL 'DKTBNAME' USING            WS-TABLE-NAME.
```

The following example forces a special table name (a View or an Alternate Index View for example) to be converted to a Data Table name. This is often done if the name is to be displayed in upper-case when printed in a message line.

```
   MOVE SPACE TO                    WS-STATUS-BYTE.
   CALL 'DKTBNAME' USING            WS-TABLE-NAME
                                    WS-STATUS-BYTE.
```


12 tablesONLINE/ISPF exit programming

The tablesONLINE/ISPF software is an optional component of the tableBASE system.

This chapter outlines the structure of a user exit program that interfaces with tablesONLINE/ISPF, shows how to write an exit, and gives examples of field, item, and table-level exit programs.

Allocating user load libraries

You may dynamically attach one or more load libraries containing user exit programs by allocating load libraries to the TBULOAD DDNAME, e.g., `ALLOC FI(TBULOAD) DSN('your.prefix.LOAD') SHR`

This ALLOC statement can be issued from:

- ISPF primary menu option 6
- native TSO
- the command line while in tablesONLINE/ISPF 4.4
- LOGON PROC

It will cause tablesONLINE/ISPF to search `'your.prefix.LOAD'` for any user exit program(s) not found in load libraries allocated to DDNAME STEPLIB. TBULOAD is searched after STEPLIB.

Structure of an exit program

User exit programs can be used to customize tablesONLINE/ISPF for your application by letting you interrupt tablesONLINE/ISPF processing with your own programs.

Note: Support for these programs is the responsibility of your company.

When a user exit program is called, tablesONLINE/ISPF processing ends and control is given to the exit program. An exit program can be invoked at the table, field, or row level. Your exit program can be called before the table is opened, before the field is edited, or before the row is edited.

The following shows the data areas that are passed between tablesONLINE and the user exit program. This information is also in the copybook, EXITISPF, in the distribution source library.

```
*-------------------------------------------------------------
* THIS IS A COBOL COPY BOOK FOR USE IN TABLESONLINE/ISPF EXIT
* PROGRAMMING. IT MAPS THE LINKAGE SECTION FOR THESE EXITS
*-------------------------------------------------------------
 01  HOOK-PARAMETER-AREA-1.
     05  HOOK-FIELD                 PIC  X(50).
     05  HOOK-FLD-DEFS              PIC  X(100).
     05  TABLE-COMMAND-AREA.
         10  TB-COMMAND             PIC  X(02).
         10  TB-TABLE               PIC  X(08).
         10  TB-FOUND               PIC  X.
         10  TB-INDIRECT            PIC  X.
         10  FILLER                 PIC  X.
         10  TB-ABEND-OVERRIDE      PIC  X.
         10  TB-ERROR               PIC S9(04) COMP.
         10  TB-COUNT               PIC S9(08) COMP.
         10  TB-LOCK-LATCH          PIC X(08).
         10  TB-ROW-OVERRIDE-LENGTH PIC S9(08) COMP.
         10  TB-ROW-ACTUAL-LENGTH   PIC S9(08) COMP.
         10  TB-FG-KEY-LENGTH       PIC S9(04) COMP.
         10  TB-FUNCTION-ID         PIC S9(04) COMP.
         10  TB-FUNCTION-AREA       PIC  X(08).
         10  TB-DATE-AREA   REDEFINES  TB-FUNCTION-AREA.
             15  TB-DATE            PIC  X(08).
         10  FILLER                 PIC  X(20).
         10  TB-RETURNED-ABS-GEN-NO PIC S9(04) COMP.
         10  TB-ERROR-SUBCODE       PIC S9(04) COMP.
     05  I-ZUSER                    PIC  X(08).
     05  TABLE-OPEN-SW              PIC  X(01).
     05  ACTION-BYPASS-IND          PIC  X(01).
     05  FIELD-COMMAND-AREA.
         10  FT-COMMAND             PIC  X(02).
         10  FT-TABLE.
             15  FILLER             PIC  X(07).
             15  FT-TABLE-SUFFIX    PIC  X.
         10  FT-FOUND               PIC  X.
         10  FT-INDIRECT            PIC  X.
         10  FILLER                 PIC  X.
```



```
           10  FT-ABEND-OVERIDE       PIC  X.
           10  FT-ERROR               PIC S9(04) COMP.
           10  FT-COUNT               PIC S9(08) COMP.
           10  FT-LOCKLATCH           PIC  X(08).
           10  FT-ROW-OVERRIDE-LENGTH PIC S9(08) COMP.
           10  FT-ROW-ACTUAL-LENGTH   PIC S9(08) COMP.
           10  FT-FG-KEY-LENGTH       PIC S9(04) COMP.
           10  FT-FUNCTION-ID         PIC S9(04) COMP.
           10  FT-FUNCTION-AREA       PIC  X(08).
           10  FT-DATE-AREA   REDEFINES  FT-FUNCTION-AREA.
               15  FT-DATE            PIC  9(08).
           10  FILLER                 PIC  X(20).
           10  FT-RETURNED-ABS-GEN-NO PIC S9(04) COMP.
           10  FT-ERROR-SUBCODE       PIC S9(04) COMP.
       05  HOOK-FILLER                PIC  X(171).
       05  I-TYPECHG                  PIC  X(01).
       05  HOOK-POINTER               PIC S9(04) COMP SYNC.
       05  DATA-ARRAY-NEW.
            10  DATA-ENTRY-NEW        OCCURS 1000
                                      PIC  X(50).
   01  HOOK-PARAMETER-AREA-2.
       05  WS-EDIT-ERR                PIC S9(04) COMP SYNC.
       05  HOOK-USER-MESSAGE          PIC  X(60).
```

The communication data area for exit programs consists of two sections:

- HOOK-PARAMETER-AREA-1 passes information from tablesONLINE/ISPF to the user exit program
- HOOK-PARAMETER-AREA-2 passes error codes and messages from the user exit program to tablesONLINE/ISPF.

The following sub-sections explain the fields of the HOOK-PARAMETER-AREA1 and HOOK-PARAMETER-AREA-2.

HOOK-PARAMETER-AREA-1

The HOOK-FIELD is a 50-byte area that stores the field content. The field content is what is displayed on the tablesONLINE Item Edit screen before you press <Enter>. For example, if an exit program is connected to the Sex field and you specify M as the Sex field value on the tablesONLINE Item Edit screen, then HOOK-FIELD would contain M for the Sex field. This field can be modified by a field exit program.

HOOK-FLD-DEFS is a 100-byte field definition that defines the attributes of HOOK-FIELD. It contains the attributes specified for the field (stored in HOOK-FIELD) when the View was originally defined:

- Display Sequence
- Key Indicator
- Field Name

- Display Length
- Display Format
- Display Attribute
- Table Location
- Table Length
- Table Format
- Edit Exit Program

This area can be mapped into the HOOK-FLD-DEFS of the EXITISPF copybook in the distribution library. For a layout of this field, please see "ITEM AREA OF VIEW TABLE" on page 361.

The TABLE-COMMAND-AREA contains the command that is to be performed when the exit program finishes processing and control is returned to tablesONLINE/ISPF. The exit program can interrogate this area to find out what tablesONLINE will do next. This area can be modified by the exit program. The distributed source file contains a copybook for the command area.

The I-ZUSER field is the eight-character identifier of the TSO user. It can be used for security validation.

TABLE-OPEN-SW is a one-byte switch that indicates whether the table identified in the TABLE-COMMAND-AREA is open when control is passed to the exit program. Values are Y and N where Y indicates that the table is open and N indicates that the table is closed.

The ACTION-BYPASS-INDICATOR controls what actions tablesONLINE performs immediately after the exit program ends. Values are Y and N where Y specifies that the next scheduled tablesONLINE/ISPF operation is to be bypassed and N specifies that the action is to be performed. The default is N. This field is set by the exit program when returning to tablesONLINE.

The FIELD-COMMAND-AREA contains the last command that was processed by tablesONLINE before the exit program started. This 72-byte area is similar to the TABLE-COMMAND-AREA.

The HOOK-FILLER area is a 171-byte area that is reserved for future use.

The I-TYPCHG field indicates whether a new item is being created (N for New) or an existing item is being updated (U for Update) by the terminal user.

The HOOK-POINTER identifies the location (count) of the field being edited. It can also be used as a subscript to view items in the following array of field values. This subscript value is identical to the count subfield value in the FIELD-COMMAND-AREA. It can be used by field exit programs to determine which field was being edited when the exit

program was invoked. One exit program, therefore, can be written for all the fields in a table if necessary. This halfword binary subscript identifies the relative number of the field being edited. Program logic can do conditional branching based on this value.

DATA-ARRAY-NEW is an array of 50-byte fields containing 1000 entries. These are tablesONLINE limitations; field values cannot exceed 50 characters and a maximum of 1000 fields are supported. This array can be viewed by field exit programs. It can be modified only by item exit programs by Indexing through individual fields.

DATA-ENTRY-NEW is a 50-byte field in the DATA-ARRAY-NEW array. There are 1000 of these fields, each of which can be accessed by indexing, using HOOK-POINTER as a subscript.

HOOK-PARAMETER-AREA-2

WS-EDIT-ERR is a halfword binary code the exit program returns to tablesONLINE. This error code appears as an exit error on the appropriate tablesONLINE screen.

The HOOK-USER-MESSAGE is a message that appears with the non-zero error code.

Writing an exit program

The tablesONLINE distribution tape includes a copybook called EXITISPF and three example user exit programs—EXAMTBLS, EXAMITMS, and EXAMFLDS. Follow the structure of the examples to assist in writing your own user exit programs.

In many applications, particularly for validations which are used often, it is worthwhile to make the exit programs themselves table-driven for ease of maintenance. For example, a table-driven exit program could validate product numbers by applying the SK (Search by Key) command against a table of products using the product number as key. If the product does not exist, then the company does not carry the product.

One advantage of a table-driven exit program is achieved by replacing SK with FK (Fetch by Key). This setup not only validates the field but also retrieves a table item. For example, if the product number is valid, you would get the item describing it. Consequently, if another part of your application needs to know the product's price or which division makes it, that information is immediately available.

Field-level exit programs

Field-level exit programs have several uses. The most common use is to validate data. Some validation is built into tablesONLINE/ISPF; these built-in checks test only syntax. If not all numbers, dates, or character strings make sense in your application, then field-level exits can do further validation. For example, an exit program can test whether a gender field contains an acceptable value by looking for either an M or an F.

When the exit program encounters invalid data, it can set the error code to inform tablesONLINE about the error. If WS-EDIT-ERR is set to a non-zero integer and the ACTION-BYPASS-IND to N, processing does not proceed until the error is corrected. Consequently, if you enter an invalid value on the Item Edit screen, all the values are not processed until the error is corrected. The terminal user may exit the screen without processing by pressing <PF3>.

In addition to validating data, field-level exit programs can check that a value is within a specified range. They can also check for valid formats in entries whose data has a fixed format, such as part numbers, serial numbers, model numbers, employee numbers, department codes, and invoice numbers. Another use of field-level exit programs is data translation. For example, an exit program can match values to titles so that a user can specify a number in the department field which points to a department name that appears in the table.

Row-level exit programs

An exit program at the row level provides more validation than at the field level. The following example illustrates how an exit program at the row level can validate a Salary field.

A field-level exit program could check that the value is a positive integer less than 200,000. This limit is too general to be very helpful in preventing errors. A row-level exit program, on the other hand, could validate pay rates against limits for each job category. Because pay rates would depend on information in another field of the row, the exit program must be in effect at the row level.

The code to implement this validation is a table lookup with Job Category as key and an upper and lower boundary as retrieved data. This system is easily maintained; if pay scales change you can update the boundary table entries and future validations will use the new scales.

Since all fields in the row are available, more complex conditions can be set up by indexing the DATA-ARRAY-NEW and HOOK-POINTER. For example, pay scales may vary with location as well as job category. This could be handled by incorporating the additional validation into the table; location, as well as category, could be part of the key.

Table-level exit programs

A table-level exit program is invoked when the table is being accessed. It specifies that an operation is performed before tableBASE processing occurs. For example, an exit program could control the users that access a certain table by checking the UserID requesting the open against a table that specifies which users can access certain tables. An additional column in the table could specify whether users have read or write authority to a table.

13 tablesONLINE/CICS exit programming

The tablesONLINE/CICS software is an optional component of the tableBASE system.

This chapter discusses the interaction between tablesONLINE/CICS and exit programs: what parameters are passed by tablesONLINE/CICS, what can be done, and what parameters are passed back to tablesONLINE/CICS. It also explains the exit programming facilities and includes a sample program. The chapter concludes with a discussion of various exit programming scenarios.

The exit programming facilities of tablesONLINE/CICS are very flexible. You may have tablesONLINE/CICS call an exit program at various times during your use of:

```
SYSTEM sign on
COMMAND LINE / LINE COMMANDS processing
    TABLE open
      ROW (ITEM) input
          FIELD input
          FIELD output
       ROW (ITEM) cross-field validation
       ROW (ITEM) output
   TABLE store
    TABLE close
SYSTEM sign off
```

Although each line above represents a logically distinct exit point, an application can use just one program for several levels of indentation above. If, for example, exits are required at Table Open, Table Store, and Table Close time, then you can write a single program that branches to perform any of the three functions and place its name in the View Supplementary Information section of the View. This reduces overhead and avoids some potential maintenance problems by allowing the three table-level operations to share common code and data definitions.

In this chapter we consider the terms row and item to be equivalent (although row is the preferred term). In addition, the term item is sometimes used to describe an independent storage area which contains table data for single row retrievals or updates. Also, the terms View and FDT are considered equivalent, although View is the preferred term. In the following sections, some tablesONLINE/CICS exit program variable names use terminology from previous releases (of tablesONLINE/CICS) for backward compatibility with exit programs developed under earlier releases.

Interacting with tablesONLINE/CICS

The primary means of communication between tablesONLINE/CICS and your exit program is through the parameters shown in Figure 13-1, all of which appear in a copybook (EXITPARM) that is supplied as part of tableBASE.

```
05  T-TBLX-PARMS.
    10  T-TBLX-PARM-INDICATORS.
        15  T-TBLX-EXIT-INDICATORS.
            20  T-TBLX-EXIT-STIF-IND    PIC X.
            20  T-TBLX-EXIT-IO-IND      PIC X.
            20  T-TBLX-EXIT-BA-IND      PIC X.
        15  T-TBLX-BYPASS-ACTION-IND    PIC X.
        15  T-TBLX-ALL-UPD-IND          PIC X.
        15  T-TBLX-KEY-UPD-IND          PIC X.
        15  T-TBLX-ITEM-UPD-IND         PIC X.
        15  T-OPER-PGM-OP-MODE          PIC X.
        15  T-TBLX-ITEM-ACTION          PIC X.
        15  T-TBLX-DUPSOK-IND           PIC X.
```

Figure 13-1: Exit program communication parameters

The tablesONLINE/CICS software sets the following parameters prior to calling the exit program:

- T-TBLX-EXIT-STIF-IND
- T-TBLX-EXIT-IO-IND
- T-TBLX-EXIT-BA-IND
- T-TBLX-ALL-UPD-IND
- T-TBLX-KEY-UPD-IND
- T-TBLX-ITEM-UPD-IND
- T-OPER-PGM-OP-MODE
- T-TBLX-ITEM-ACTION
- T-TBLX-DUPSOK-IND.

tablesONLINE/CICS derives the value of these parameters from either the original definition of the View or events triggered by the usage of tablesONLINE/CICS.

The parameter T-TBLX-BYPASS-ACTION-IND is specified when the program is ready to return to tablesONLINE/CICS. In certain circumstances, for example, before returning to tablesONLINE/CICS, you may also change the parameters:

- T-TBLX-BYPASS-ACTION-IND
- T-TBLX-ALL-UPD-IND
- T-TBLX-KEY-UPD-IND
- T-TBLX-ITEM-UPD-IND

STIF indicator

The STIF indicator (T-TBLX-EXIT-STIF-IND) tells the exit program from what high-level processing point it is being called.

Legal values are:

- **S** for System security level
- **C** for Command processing level
- **T** for Table level
- **I** for Item (row) level
- **F** for Field level

Specify these exit points when defining a View in tablesONLINE/CICS.

A STIF indicator of **S** indicates a system-level exit, typically used for security purposes. This case will be ignored here; for security information, see the *tableBASE Administration Guide*.

An exit program can be called from more than one place in tablesONLINE/CICS. It is possible to write a single program that branches on the STIF indicator to perform both table and row-level functions or to write one exit that branches on table name to provide different validations for different tables.

The highest-level control structure of a tablesONLINE/CICS user exit program is nearly always a multi-way branch. In some applications, the STIF indicator controls the branching. In other applications a combination of indicator bytes is used. An exit program that needs to behave differently for different tables may branch first on table name and then, for each table, branch on the indicators.

I/O indicator

The parameter T-TBLX-EXIT-IO-IND identifies the type of I/O operation (Input data, Output data, Open table, Store table, etc.) or other tablesONLINE/CICS processing being performed at the time the exit was invoked. In this context, input refers to any movement of data along a route beginning from a tableBASE library and ending at the display screen. Similarly, output implies any movement of data along a route beginning from the display screen and ending with a tableBASE library. This includes movement between intermediate locations, including between tables in the TSR and internal item/row areas, or between item/row areas and individual fields in display buffers.

This indicator has the values listed in Table 13-1.

Table 13-1: I/O indicator

STIF	I/O Ind	Meaning
C - Cmd	M	Menu Dialogue Command
C	I	Identify Table Object Command
C	S	Select Item/Row in Table Command
C	E	Edit Item/Row Command
T - Table	O	Open
T	S	Store
T	C	Close
I - Item	I	Input—table to row buffer
I	D	Delete
I	M	Move
I	X	Edit Cross Fields
I	N	New—row buffer to table
I	U	Update—row buffer to table
I	Q	Quit
F - Field	I	Input—row buffer to field
F	O	Output—field to row buffer

Before/After indicator

The T-TBLX-EXIT-BA-IND parameter denotes whether the exit is being called Before or After (values **B** or **A** respectively) the operation described above (that indicated by the I/O indicator) is performed. This value is specified when you are defining a View in tablesONLINE/CICS.

Exit programs should treat these three indicators (STIF, I/O, Before/After) as strictly read-only data. They can be accessed either individually or as a three character string.

Bypass Action indicator

In T-TBLX-BYPASS-ACTION-IND (Table 13-3 on page 338) you tell tablesONLINE/CICS what actions should be taken when the exit program returns to tablesONLINE/CICS.

Legal values are:

- **blank** (OK)
- **E** (Error)
- **W** (Warning)
- **I** (Information)
- **Y** (Yes, bypass next tablesONLINE/CICS action).

The indicator is initialized to blank when the software enters the exit program. If exit program code sets up a message but does not set the T-TBLX-BYPASS-ACTION-IND parameter then the NORMAL-EXIT routine will move the TYPE from the message table into T-TBLX-BYPASS-ACTION-IND.

The **Y** code has unique uses for different indicator combinations, as it applies to the next action expected at various points in processing. This code should only be used as an explicit action code, not as a message TYPE.

The right hand column in Table 13-2 on page 333 shows the action tablesONLINE/CICS will take if the exit program returns with the BYPASS-ACTION-IND set to **blank** (indicating normal completion) or to **W** or **I** (indicating that a Warning or an Information message has been generated by the exit program).

Update indicators

Before calling the exit program, tablesONLINE/CICS sets three flags to indicate what data the user has changed. The flags affect tablesONLINE/CICS actions in later processing.

Referring to the parameter list under "Interacting with tablesONLINE/CICS" on page 326, the T-TBLX-ALL-UPD-IND parameter is set to **Y** whenever any *row* on the table has been changed. This affects whether a new generation is stored when the terminal user leaves the editor.

The parameter T-TBLX-KEY-UPD-IND is set to **Y** if any *key* field of the current row has been changed; this affects how an item-area-to-table-data move is done. If the key has not changed, and the multi-user update is not in effect, tablesONLINE/CICS can use a replace-by-count (RC) command. If the key has changed, tablesONLINE/CICS has more to do.

The parameter T-TBLX-ITEM-UPD-IND is set to **Y** if any *field* has been changed. The flag affects whether an item-area-to-table move is done at all.

You will not normally change these fields, except in special cases. For example, an exit program which updates a field the terminal user has not touched, such as a Date Last Accessed field, should set T-TBLX-ITEM-UPD-IND to **Y**.

Program Operating Mode

T-OPER-PGM-OP-MODE is known as the PROGRAM OPERATING MODE flag. It tells your program whether tablesONLINE/CICS was operating in **B** (Browse) or **E** (Edit) mode when the exit was invoked.

This is commonly used in an exit program called with an exit taken before a table is opened; it is used for security or integrity checking. If the flag contains **E** (Edit), then the table will be opened for write access. If the flag contains **B** (Browse), then the table will be opened in read-only mode.

Item Action flag

The T-TBLX-ITEM-ACTION flag is determined when the editor is invoked and should be treated as read-only. Typically, it is identical to the I/O indicator, T-TBLX-EXIT-IO-IND. If the terminal user keys in **U** for Update in selecting a row, then decides to create a new row and keys in **NEW** on the command line, the user's intent changes and these flags will be different. In this case the exit program will be provided with ITEM-ACTION **U** for Update and IO-IND **N** for New.

Duplicate Keys Allowed indicator

DUPSOK-IND is the Duplicate Keys Allowed indicator from the View. It is read-only.

IXF Exit indicator

The IXF Exit indicator combination is used for handling cross-field or whole item (row) data validation. The tablesONLINE/CICS program calls an exit program with these indicators before taking any action which outputs that item (row) data, that is, before any Update or New operation. This does not happen before Delete, which does not output the item, nor before Move which does not alter fields. The actual output operation is attempted only if cross-field validation succeeds. The IXF user exit is also invoked when the ENTER or the EXECUTE keys are pressed. This allows a user to validate all the field relationships by pressing <Enter>.

For the application developer, the IXF exit is one of the most powerful and convenient tools tablesONLINE/CICS provides. It is called after all field-level validations have run, but before the table updating action is taken, so it provides a central location for a variety of data-control functions. See "Sample exit program" on page 341 for a variety of IXF examples.

tablesONLINE/CICS's next action on a successful return from the IXF exit can vary. If the terminal user is attempting to operate on the item, then the next action will be an exit program call with indicators IUB, INB, or IMB. If the terminal user has just hit <Enter> to get cross-field validation without attempting any further action, then the next action tablesONLINE/CICS takes will be to return control to the user.

A BYPASS-ACTION-IND value of **Y** returned from an IXF exit prevents any action which would move the user off the current item. The user can update the row currently displayed but operations such as getting the next or previous row will always fail. This is useful for data integrity purposes. It is used, for example, in many of the tablesONLINE/CICS utility programs.

Here are possible values for the indicator and their associated meanings:

BYPASS-ACTION-IND:

- **Y**—Data valid but moving off the row is forbidden
- **E**—Bad data
- **W**, **I**, or **space**—Good data

IQU exit

The indicator combination IQU is another special case, provided for use when the terminal user backs out of an action with the Cancel key <PF12> or leaves an item without having changed any of its data. If other exits use enqueuing at entry to items for instance, it is necessary to have code in this exit to dequeue items when the user abandons the update.

The IQU exit is invoked when the T-TBLX-ITEM-UPD-IND is set to **N** by tablesONLINE/CICS, indicating that the terminal user is leaving an item without making changes. When the IQU indicator appears, both the IUB and the IUA exits are not invoked since these combinations are mutually exclusive.

The indicator T-TBLX-ITEM-UPD-IND is set to **Y** by tablesONLINE/CICS when any field is modified. If modifications are made in exit programs and not by some user action detectable by tablesONLINE/CICS, then this indicator must be set to **Y** in the exit program in order for updates to be applied.

For the IQU exit, the user is trying to abandon the current row, leaving it unchanged. In this case the BYPASS-ACTION-IND options are:

- **Y**—Not used
- **E**—Remain on row
- **W**, **I**, or **space**—Abandon row and continue per user's instructions

Note: Under special circumstances updates to table items must be postponed until the invocation of an IUB exit. It is possible to force the invocation of the IUB and IUA exits. This may be done by setting the T-TBLX-ITEM-UPD-IND to **Y** in a prior exit. Usually this is done in the IXF exit after a successful edit.

To simplify the explanation of the flow of control, the exits have been grouped into four categories below: Before Action Exits, After Action Exits, System Exits, and Command Line Exits.

Top-level control structure for an exit

The typical top-level control structure for a tablesONLINE/CICS user exit program is based on a multi-way branch depending on the indicator bytes in T-TBLX-EXIT-INDICATORS.

Table 13-2 lists all possible values of T-TBLX-EXIT-INDICATORS. For each combination of level (system, command, table, item, and field) Input/Output, Operation and Timing, the table summarizes the tablesONLINE action.

Table 13-2: Exit timing and indicator combinations

Indicator combination	Level	Input/ output operation	Timing	tablesONLINE/CICS action on normal return
SIB	System	Sign on	Before	TBOLACT Validation
SIA	System	Sign on	After	
SOB	System	Sign off	Before	Return to CICS
SOA	System	Sign off	After	SOB Exit
CMB	Cmd Line	Menu	Before	Parse Menu command line
CMA	Cmd Line	Menu	After	Parse Menu Item selection
CIB	Cmd Line	Identify	Before	Parse Identify command line
CSB	Cmd Line	Edit table	Before	Parse Edit Table command line
CSA	Cmd Line	Edit table	After	Parse Edit Table line cmd (pair)
CEB	Cmd Line	Edit row	Before	Parse Edit Row command line
TOB	Table	Open	Before	Open table (OR or OW)
TOA	Table	Open	After	
TSB	Table	Store	Before	Store table (ST)
TSA	Table	Store	After	
TCB	Table	Close	Before	Close table (CL)
TCA	Table	Close	After	
IIB	Item (row)	Input	Before	Fetch item (FC or FK)
IIA	Item (row)	Input	After	

Table 13-2: Exit timing and indicator combinations (Continued)

Indicator combination	Level	Input/ output operation	Timing	tablesONLINE/CICS action on normal return
IUB	Item (row)	Update	Before	Update item (RC, RK or combinations using DK, DC, IK and IC)
IUA	Item (row)	Update	After	
IMB	Item (row)	Move	Before	Move item (combination using DK, DC, IK and IC)
IMA	Item (row)	Move	After	
IDB	Item (row)	Delete	Before	Delete item (DK or DC)
IDA	Item (row)	Delete	After	
INB	Item (row)	New	Before	Create item (IK or IC)
INA	Item (row)	New	After	
IXF	Item (row)	any	Output	See "IXF Exit indicator" on page 331 and
IQU	Item (row)	abort	Update	"IQU exit" on page 332
FIB	Field	Input	Before	Move data from item to screen
FIA	Field	Input	After	
FOB	Field	Output	Before	Move data from screen to item
FOA	Field	Output	After	

Before Action exits

When T-TBLX-EXIT-BA-IND (the Before/After INDicator in the EXITPARM copybook structure) is **B**, the exit program is being called before some specified action is executed, and on normal return tablesONLINE/CICS will take that action.

In most applications, these Before exit programs (TOB, TCB, TSB, IIB, IUB, INB, IMB, IDB, FIB, and FOB), validate data to help ensure that all will go well when tablesONLINE/CICS performs the action.

Data validation

For data validation exit programs called Before the action, the BYPASS-ACTION-IND may be set to E, W, I, or space to indicate whether or not the specified tablesONLINE/CICS action in Table 13-2 on page 333 should be executed:

- **Y**—Not used
- **E**—Error, tablesONLINE/CICS should not continue with the action
- **W**, **I**, or **space**—tablesONLINE/CICS should attempt the action

Replacing tablesONLINE/CICS action

You may also write Before exit programs which replace the tablesONLINE/CICS actions with their own. Some examples are discussed in the applications section below. Such programs set T-TBLX-BYPASS-ACTION-IND to **Y** to prevent tablesONLINE/CICS from taking an action specified in Table 13-2 on page 333.

Here are the possible values of the BYPASS-ACTION-IND indicator and corresponding actions:

- **Y**—Action successfully completed by exit program
- **E**—Action could not be completed
- **W**, **I**, or **space**—Not used

When the exit program returns to tablesONLINE/CICS in both of the above situations (i.e., BYPASS-ACTION-IND is **Y** or **E)** tablesONLINE/CICS will do one of three things:

- continue with the action if **W**, **I,** or **space** has been set in BYPASS-ACTION-IND
- continue but skip the action if **Y** has been set in BYPASS-ACTION-IND
- go back to ask for better data if **E** has been set in BYPASS-ACTION-IND

After Action exits

When T-TBLX-EXIT-BA-IND is **A,** no action is shown on the right in Table 13-2 on page 333 (except for SOA, CMA and CSA). The exit program is being called after some action has been completed. The next thing tablesONLINE/CICS does is to await user input from the terminal. In most cases, this is sufficient.

Sometimes it is necessary to prevent tablesONLINE/CICS from giving control back to the user. Consider an exit program which examines the row just retrieved and finds that it should not be displayed. The exit program could do its own looping to retrieve another row, but there is an alternative. If an IIA or FIA exit program sets BYPASS-ACTION-IND to **Y**, tablesONLINE/CICS bypasses the return of control to the user and loops around to try the action again with the next field or row. Such a loop continues until the exit program returns **blank**, **W**, **I,** or **E** for Error, or until tablesONLINE/CICS runs out of fields or rows.

Here are the possible values of the BYPASS-ACTION-IND indicator and corresponding action for exit programs called After Item or Field Input, (IIA, FIA):

- **Y**—Excluded data, get next without displaying
- **E**—Nothing retrieved, re-display old data
- **W**, **I**, or **space**—Retrieved data, display it

For any other After exit, a T-TBLX-BYPASS-ACTION-IND of **Y** would be meaningless and has no effect except for the TOA exit.

TOA exit

There are several functions that tablesONLINE/CICS performs after opening a table. These functions can be bypassed if this indicator, BYPASS-ACTION-IND, is set to **Y**. If this is done, care must be taken to perform in the exit program any or all of the functions described in the following paragraph.

After opening a table, tablesONLINE/CICS performs the following functions:

- Gets the Definition to determine: Row Size, Organization, and Index type (True or Pointer);
- If necessary, either reorganizes the table into sequential order with a binary search or Invokes an Alternate Index for a non-sequential Indexed table. No action can be taken when the table is being browsed;
- Verifies that the Row Size, Key Size, and Key Location of the table match those of the View.

For all remaining exit programs called After any action but input (TCA, TSA, IDA, IMA, INA IUA, FOA), the possible values for the BYPASS-ACTION-IND indicator are:

- **Y**—Not used, re-display

- **E**—Nothing retrieved, display it
- **W**, **I**, or **space**—Have data, old data, display it

tablesONLINE/CICS system exits

System-level exits are used to provide an opportunity for special user validation and to set system defaults.

Note: These are not the same as tableBASE System or User Exits.

For the system sign-on exits, SIB and SIA, tablesONLINE/CICS system initialization has not been completed. Message stacks, for example, are not yet established, so standard tablesONLINE/CICS message processing is unavailable. In the case of SIB, the exit, which is specified in TBOLCNST, controls the access to tablesONLINE/CICS. The SIA exit occurs after TBOLACT application control processing is completed; this exit can be used to perform tracking of user-specific usage of tablesONLINE/CICS. Setting the BYPASS-ACTION-IND to **Y** or **E** prevents a user from accessing tablesONLINE/CICS.

For the System sign off exits, SOB and SOA, tablesONLINE/CICS system initialization has been completed. The SOB is the logical counterpart to the SIA exit. If a timer was activated at the SIA exit, here would be the place to inspect the timer and record if so desired. Setting the BYPASS-ACTION-IND at this time has no effect.

Command Line/Line Command processing exits

Command-level exit points exist for **Menu**, **Identify Table**, **Edit/Browse Table**, and **Edit/Browse Row** screens.

Setting the BYPASS-ACTION-IND in any of the these Command-level exits has no effect; processing will continue as if the exit was not invoked.

Command Line exits

The indicator combinations of CMB, CIB, CSB, and CEB handle pre-processing of Command Line commands entered at the top of the **Menu**, **Identify Table**, **Edit Table**, and **Edit Row** screens, respectively. Control is transferred to the exit program after the command line contents have been parsed and translated (using the TBOLCMDS translation table) into T-COMMAND and T-CMD-PARM before they are executed by tablesONLINE/CICS. This is useful for validating user authorization for particular tablesONLINE/CICS commands, enhancing the operation of tablesONLINE/CICS commands, or for creating entirely new commands.

If you are processing commands that are foreign to tablesONLINE/CICS, you must set the T-COMMAND field to spaces or any valid tablesONLINE/CICS command to avoid the generation of an invalid command error message.

Line Command exits

Line Command processing is available only for **Menu** and **Edit Table** screen processing. These exits are invoked after a menu item has been selected, or a set of rows has been selected using the line commands.

The Menu Line Command processing exit, CMA, may be used for automatic tailoring of menu transfers required for security considerations or for adapting to dynamic events such as system configuration changes. Control is transferred to the exit directly after the user has selected a valid menu item for processing and prior to the transfer of control to the program as specified in the selected menu item. The selected menu item is available in T-MENU-TABLE-ITEM for inspection, as well as modification.

The Edit Table Line Command processing exit, CSA, receives control after a line command at the left hand column of the screen has been validated by tablesONLINE/CICS but before the line command is executed. A valid line command is a single character associated with a single row in a table or a double character pair of line commands bound to a series of rows in the table. The CSA exit is invoked once for each valid line command on the screen. This exit point may be used to enhance the operation of existing line commands, or to create entirely new line commands. Three Line Command symbols can be defined that are distinct from the line command symbols currently in use by tablesONLINE/CICS. The default values are set in the TBOLCNST table and have the following field names: R LINE CMD, C LINE CMD, and X LINE CMD.

BYPASS-ACTION-IND summary

Table 13-3 below summarizes the meaning of the bypass indicator for various exit types.

Table 13-3: Bypass Action indicator summary

Exit type	**Return Bypass Action indicator**		
	Y	**E**	**W, I, or space**
System before action (SIB, SIA)	Abandon sign on	Abandon sign on	Not used
System after action (SOB, SOA)	Not used	Not used	Not used

Table 13-3: Bypass Action indicator summary (Continued)

Exit type	Return Bypass Action indicator		
	Y	**E**	**W, I, or space**
Command (CMB,CMA, CIB, CSB, CSA, CEB)	Not used	Not used	Not used
Data validation before action (TOB, TCB, TSB, IDB, FIB, and FOB)	Not used	Error, tablesONLINE should not continue the action	tablesONLINE should attempt the action
Intervening before to replace tablesONLINE action (TOB, TCB, TSB, IIB, IUB, INB, IMB, IDB, FIB, FOB)	Action successfully completed by exit program	Action could not be completed by exit program	Not used
After item or field input (IIA, FIA)	Excluded data, get next without displaying	Nothing retrieved, re-display old data	Retrieved data, display it
After Table Open (TOA)	See "TOA exit" on page 336	Error, tablesONLINE should not continue with the action	Not used
All other after exits (TCA, TSA, IDA, IMA, INA, IUA, FOA)	Not used redisplay	Nothing obtained, re-display old data	Got data, display it
Cross-field data validation (IXF)	Data valid but moving off the row is prevented	Bad data redisplay	Good data
Quitting (IQU)	Not used	Remain on row	Abandon item and continue per user's instructions

Constructing an exit program

The tablesONLINE/CICS distribution tape in the education dataset includes a sample user exit program called EXITPGMC and two copybooks called EXITPARM and EXITWS. These are a good starting point for developing your own exit programs. Different versions of the exit programs with some detail removed and comments added are listed in "Sample exit program" on page 341.

The recommended control structure of an exit program includes the following sections:

- Initialization of system variables
- Initialization of user variables
- Test for operational integrity (defined below)
- Multi-way branch on EXIT-INDICATORS
- User code for each branch where action is required
- Exit for INVALID CALL
- NORMAL EXIT, with or without message to user.

Operational integrity

Tests of operational integrity should be included in an exit program. For example, if the program contains table-specific code, then it cannot be expected to handle data from some other table.

An essential integrity rule is that the table name in the call T-TBLX-TABLE must match the one the program was designed for. It should test this and quit if the test fails rather than continue, and possibly wreak havoc, with mismatched code and data. To quit, it could either branch to INVALID-CALL or to a user routine which sets up a message with message type E then goes to NORMAL-EXIT. The latter method is preferable since it provides better messages.

Control structure

The most common control structure here is the multi-way branch on the EXIT-INDICATORS. For indicator combinations which you wish to handle, branch to your own labels. If there are combinations for which no action is needed, branch directly to NORMAL-EXIT. Catch anything that falls through those tests with an unconditional branch to INVALID-CALL.

Below each application-specific label comes your application code for handling each case. This code should not branch to INVALID-CALL; it is too late for that. All these routines should end with an unconditional branch to NORMAL-EXIT unless you wish to write your own routine for returning control to tablesONLINE/CICS.

Sample exit program

The following is the skeleton of a user exit program segmented into discrete pieces with explanatory text added for each piece. This program, or one similar to it, is available to customers in the xxx.TBASE.SRC file (see member EXITPGMC).

This is the framework code which all exits will contain. The example (see Table 13-4 on page 368) omits the logic which does the actual work. Program code is in upper-case, while lower-case or mixed-case are used for the text and for lines within the code which the user must supply or alter.

```
 IDENTIFICATION DIVISION.
*
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 01  H-EXIT-DUMP-ID.
     05  FILLER         PIC X(26) VALUE 'TABLESONLINE EXIT PROGRAM '.
     05  H-PROGRAM      PIC X(8)  VALUE 'EXITPGM'.
     05  FILLER         PIC X(2)  VALUE '--'.
     05  H-COMPILED     PIC X(8)  VALUE SPACES.
     05  FILLER         PIC X(20) VALUE ' WORKING STORAGE --'.
*
```

The code presented above is essentially housekeeping information. The program will compile and run with any syntactically correct values.

Out of date information accumulates here during debugging and maintenance work.

Note: DataKinetics recommends that you keep this information up to date.

```
     *
          COPY EXITWS.
     *
```

The copybook, EXITWS, documented after this program is recommended for all exit programs. It provides a work area for the interface between the exit program and tablesONLINE/CICS.

```
****************************************************************
*    USER WORKING-STORAGE
****************************************************************
```


User working storage can be whatever is required, but will often include something like the following:

```
*
 01  W-XXXX-COMMAND-AREA.
     05  W-XXXX-COMMAND               PIC XX   VALUE SPACES.
     05  W-XXXX-TABLE                 PIC X(8) VALUE 'XXXX'.
     05  W-XXXX-FOUND                 PIC X    VALUE SPACES.
     05  W-XXXX-INDIRECT-OPEN         PIC X    VALUE LOW-VALUES.
     05  RESERVED                     PIC X    VALUE LOW-VALUES.
     05  W-XXXX-ABEND-OVERRIDE        PIC X    VALUE SPACES.
     05  W-XXXX-ERROR                 PIC S9(4) COMP VALUE +0.
     05  W-XXXX-COUNT                 PIC S9(9) COMP VALUE +0.
     05  W-XXXX-LOCK                  PIC X(8) VALUE SPACES.
*   Release 5.x/6.0 command area extension
     05  W-XXXX-ROW-OVERRIDE-LENGTH   PIC S9(9) COMP VALUE +0.
     05  W-XXXX-ROW-ACTUAL-LENGTH     PIC S9(9) COMP VALUE +0.
     05  W-XXXX-FG-KEY-LENGTH         PIC S9(4) COMP VALUE +0.
     05  W-XXXX-FUNCTION-ID           PIC S9(4) COMP VALUE +0.
     05  W-XXXX-FUNCTION-AREA         PIC X(28) VALUE LOW-VALUES.
     05  W-XXXX-DATE-AREA             REDEFINES W-XXXX-FUNCTION-AREA.
         10  W-XXXX-DATE              PIC X(8).
         10  RESERVED                 PIC X(20).
     05  W-XXXX-RETURNED-ABS-GEN-NO   PIC S9(4) COMP VALUE +0.
     05  W-XXXX-ERROR-SUBCODE         PIC S9(4) COMP VALUE +0.
*
 01  W-XXXX-ROW-AREA.
         10  field area declarations.
*
```

Note: DataKinetics recommends that you use the automatically generated copybook facility found in tablesONLINE/CICS. This will ensure a consistent naming convention between the online interface and the program names, as well as a reduction in testing and maintenance.

A tableBASE command area and a row area are declared so that the exit program can use tableBASE services without affecting other parts of tablesONLINE/CICS. Routines below will initialize these areas by copying the values from the corresponding tablesONLINE/CICS areas on entry, use them for various operations within the exit program, and in some cases, copy some data back to the tablesONLINE/CICS areas before returning.

```
 LINKAGE SECTION.
****************************************************************
*
*    DFH EXECUTIVE INTERFACE BLOCK.
*
****************************************************************
*
*    DFH COMMUNICATIONS INITIALIZED AT ARRIVAL FROM CICS.
*
*    THIS IS THE DFHCOMM AREA FOR EXIT PROCESSING
*
****************************************************************
*
 01  DFHCOMMAREA.
     05  D-EXIT-PARM-POINTER              POINTER.
     05  D-EXIT-ITEM-POINTER              POINTER.
     05  D-EXIT-FIELD-POINTER             POINTER.
*
 COPY EXITPARM.
*
```

This code is required for addressability of the parameters that the tablesONLINE/CICS system will pass to the exit program. It must be copied verbatim into all exit programs. EXITPARM, the copybook for the parameters passed to every exit program by tablesONLINE/CICS, is documented after this program.

```
****************************************************************
*
*    TABLE ITEM AREA - USED FOR VALIDATION.
*
****************************************************************
*
 01  L-TBLX-TABLE-ITEM.
   03  L-WORKING-ITEM                  PIC X(512).
*
```

This area must be declared large enough to handle the largest table row the exit program will ever have to deal with. The tableBASE limit on row size is 32,767 bytes.

```
****************************************************************
*    TABLE xxxxxxxx ROW AREA
****************************************************************
*
   03  L-xxxxxxxx-ROW-AREA              REDEFINES L-WORKING-ITEM.
     10  whatever.
*
```


Having defined this large working block, then define a data area within it suitable for the table actually being dealt with. In some exits there might be several redefinitions here to deal with rows from different tables or different row layouts within a single table.

```
*****************************************************************
*    FIELD IN MAP AREA.
*****************************************************************
*
 01  L-MAP-FIELD-DATA.
   03  L-MAP-FLDATA-L                  PIC S9(4) COMP.
   03  L-MAP-FLDATA-A                  PIC X.
   03  L-MAP-FLDATA-X                  PIC X(4).
   03  L-MAP-FLDATA                    PIC X(51).
```

This defines a field of the screen map tablesONLINE/CICS uses. The four parts of such a definition are length, display attribute, extended attribute, and the display data.

Users may wish to redefine L-MAP-FIELD-DATA to suit their application. Some caution is necessary here; the area's size is fixed and attempting a redefinition which increases that size may have unpredictable results. L-MAP-FIELD-DATA has space for 50 characters of text plus a field terminator attribute, “skip to next field”. Beyond that are some tablesONLINE/CICS internal areas which should not be overwritten.

```
 PROCEDURE DIVISION.
*****************************************************************
*
 A000-INITIALIZATION-SECTION.
*
*****************************************************************
*
     MOVE WHEN-COMPILED TO EXIT-COMPILED.
*
```

This line supports debugging by copying the compilation time so that it will appear in a dump.

```
*****************************************************************
*                                                               *
*    EXIT PROGRAM LINKAGE SETUP & INITIALIZATION                *
*                                                               *
*****************************************************************
*
*    SETUP EXIT PROGRAM LINKAGE
*
     SET ADDRESS OF T-TWA TO D-EXIT-PARM-POINTER.
*
     IF  D-EXIT-ITEM-POINTER   NOT = NULL
         SET ADDRESS OF L-TBLX-TABLE-ITEM TO D-EXIT-ITEM-POINTER.
*
     IF  D-EXIT-FIELD-POINTER  NOT = NULL
         SET ADDRESS OF L-MAP-FIELD-DATA TO D-EXIT-FIELD-POINTER.
*
```

This code sets up addresses on which much else will depend. It must be copied verbatim into all exit programs.


```
*    EXIT PROGRAM INITIALIZATION
*
     MOVE SPACES                  TO  T-TBLX-BYPASS-ACTION-IND.
     MOVE ZERO                    TO  T-DSPL-CONV-FIELD-ERROR.
*
```

This initializes the areas which higher-level tablesONLINE/CICS code treats as returned values from an exit program.

```
*
     MOVE SPACES                  TO  T-MSGX-KEY.
     MOVE ZERO                    TO  T-MSGX-INSERT1-LENGTH.
     MOVE ZERO                    TO  T-MSGX-INSERT2-LENGTH.
```

This clears the message area. The NORMAL-EXIT routine will attempt to display anything in this area which is not spaces.

```
*    SAVE ENVIRONMENT (DEBUGGING & RESTORING BEFORE RETURN)
*
     MOVE T-TBLX-COMMAND-AREA     TO  H-COMMAND-AREA.
*
```

Here the tablesONLINE/CICS command area is copied to the area that the tablesONLINE/CICS exit program interface code will use. This is necessary so that the exit program can use tableBASE commands without altering command areas which other parts of the system rely on. This area needs to be initialized so the exit will have the correct count and lock key values for access to the table tablesONLINE/CICS is working with.

```
*
     MOVE 'LL'                    TO  H-COMMAND.
     CALL 'TBLBASE'    USING          T-TRPARM
                                      H-COMMAND-AREA
                                      H-LIB-LIST.
*
```

This saves the library concatenation list so that it too can be restored to its original state before returning to tablesONLINE/CICS. This code, and the related restoration code, must be included in any exit which will alter the library concatenation list.

```
*
     MOVE 'LS'                    TO  H-COMMAND.
     CALL 'TBLBASE'    USING          T-TRPARM
                                      H-COMMAND-AREA
                                      H-STATUS-SAVE.
*
```

This saves the current status switches so that they may be restored to their original state before returning to tablesONLINE/CICS.

In Version 6 all the example code uses a new TBPARM with every tableBASE call. Currently, your existing exits will have T-TBPARM coded. Because tablesONLINE now supports the ability to edit a table in a Read/Write VTS-TSR, this coding has now become

T-TRPARM as you see above. Whether you need to change your existing exits is explained later (see "The EXITPARM copybook" on page 353).

```
*****************************************************************
*
*    EXIT PROGRAM SPECIFIC INITIALIZATION
*
*****************************************************************
```

Whatever is required by way of initialization for this particular exit program goes in here. One of the more important issues to consider is whether the exit program will be making any calls to tableBASE itself.

tablesONLINE/CICS expects its command area to be returned as if the exit program were not present and only the expected tableBASE commands had been executed. Developers must ensure that this assumption is valid for their programs. The way to guarantee that the tablesONLINE/CICS command area is unchanged is to have the exit program use its own command area.

```
*
     MOVE T-TBLX-COMMAND-AREA     TO  W-xxxxx-COMMAND-AREA.
*
```

Here the tablesONLINE/CICS command area is copied to the area that the tablesONLINE/CICS exit program interface code will use. This is necessary so that the exit program can use tableBASE commands without altering command areas which other parts of the system rely on. This area needs to be initialized so the exit will have the correct count and LOCK-LATCH password for access to the table that tablesONLINE/CICS is working with.

In some cases, the exit program requires write access to the table tablesONLINE/CICS is using. For example, an exit that works on a personnel table might update the employee count field of a supervisor's record when it creates a record for a new person reporting to that supervisor. In a case like this, the exit will need the table name and lock key information from the tablesONLINE/CICS command area. The code above sets this up.

Note: If write access to the table is not required, then the LOCK-LATCH password is not required. In this case, the LOCK-LATCH password should be set to spaces immediately after the line above, or the code above should be modified to move only required fields such as table name rather than the entire command area.

There may be instances where the exit's function is to modify the behavior of a tableBASE command. For example, to make Get Next return the next row which this user is allowed to access rather than just the next row on the table, the developer must ensure that the tablesONLINE/CICS command area T-TBLX-COMMAND-AREA is returned in an appropriate state.

This is achieved by issuing whatever calls are needed using the user command area. On success, the count can be copied to the tablesONLINE/CICS command area so that tablesONLINE/CICS sees results matching the data it is getting. On failure, the

tablesONLINE/CICS command area is returned largely as it was passed but with some carefully chosen fields updated. The exit might, for example, set the found indicator to N and/or adjust the count field. The necessary time should be spent here to ensure that the code handles all cases appropriately.

In an exit program which does not issue tableBASE commands — for example checking that certain field values in the current row meet certain conditions — you can simply remove the code above and the data area definitions supporting it.

It is also possible to write exits that alter the tablesONLINE/CICS command area to achieve specific effects. Please consult tableBASE customer support if this appears necessary for your applications.

```
*****************************************************************
*                                                               *
*    TABLE / OP-MODE DEPENDENT EXIT PROCESSING                  *
*                                                               *
*    SELECT EXIT ACTION BY EXIT INDICATORS                      *
*                                                               *
*****************************************************************
```

At this point the business logic of the exit program starts. Typically, use some form of multi-way branch on the three indicator bytes which show how the exit has been called. For example, TOB in these bytes show that the exit was called:

- **T**—from Table level
- **O—**for the Open operation
- **B**—Before the operation

Details will vary widely with application and developer. The branch on indicators may be done as one large branching structure using the 3-byte string or as nested structures with each byte handled at a different level.

In many applications, there will also be validation and branching on table name. Again, details will vary greatly. The table name may be examined before the indicators, and a different indicator test used for each table, or the table test may be nested under the indicators test so that different tables can be checked in each indicator case.

All possible states must be accounted for. Build the conditional structure so that the cases you expect to handle branch to the appropriate parts of your code and all other cases branch to the Y200-INVALID-CALL routine or to some similar error routine.

Typically, then, the code here will consist of some multi-way branching structure at the top level and other specific routines required to handle the various cases. Each of these will terminate by executing one of the routines on the following pages.

User code can branch to one of two labels when leaving an exit program: Y100-NORMAL-EXIT for normal exits, with or without a message to the user, or to Y200-INVALID-CALL.

It is possible to replace one or both routines with your own code. However, the basic operation must be consistent with the code provided below.

```
 Y100-NORMAL-EXIT.
*****************************************************************
*
*    EXIT PROGRAM RETURNS TO CALLER AFTER RESTORING TBCALL COMMAND
*
*****************************************************************
*
*    RESTORE LIBRARY CONCATENATION ORDER  &  TURN ABEND ON
*
     MOVE 'ML'                    TO H-COMMAND.
     CALL 'TBLBASE' USING T-TRPARM
                          H-COMMAND-AREA
                          H-LIB-LIST.
*
     MOVE 'CS'                    TO H-COMMAND.
     CALL 'TBLBASE' USING T-TRPARM
                          H-COMMAND-AREA
                          H-STATUS-SAVE.
```

The statements above restore the library concatenation list to its saved state and reset the status switches to their saved state. tablesONLINE/CICS expects its library list to remain unaltered and abend processing to be enabled at all times, so any exit program that modifies the list or disables the abend status switch must restore them as above.

Once the library list and status switches have been restored, a message may be sent if there is one:

```
*
     IF  T-MSGX-KEY  NOT =  SPACES
         GO TO  Y700-SEND-MESSAGE.
*
```

Otherwise return control to tablesONLINE/CICS.

```
*
     GO TO  Y900-RETURN-TO-CALLER.
*
```

User code should branch here for all invalid parameter settings, that is, for any parameters that indicate a mismatch between the exit program's design and its usage in some View.

If you expect your program to be called only as an item-level exit, then branch here for other settings of the STIF indicator. If you expect your program to be used for only one table, branch here for other table names.

```
*
 INVALID-CALL.
*
*    SETUP MESSAGE - ERROR IN CALLING EXIT - INVALID INDICATORS
*
     MOVE user-invalid-code         TO  T-MSGX-KEY.
     MOVE 8                         TO  T-MSGX-INSERT1-LENGTH.
     MOVE H-PROGRAM                 TO  T-MSGX-INSERT1-VALUE.
     MOVE 12                        TO  T-MSGX-INSERT2-LENGTH.
     MOVE T-TBLX-PARM-INDICATORS    TO  T-MSGX-INSERT2-VALUE.
     GO TO  Y700-SEND-MESSAGE.
*
```

All this code does is set up a message showing the exit program name and the indicator bytes, then goes to the code which sends the message.

This bypasses the restoration code used in Y100-NORMAL-EXIT. The assumption is that bad parameters will be noticed early, that user code will branch here before there are any undesired side effects that need to be backed out. If your code does not detect invalid parameters until after it has made significant changes, then it is not safe to use this route back to tablesONLINE/CICS.

The possibilities then are:

- rewrite your code to test for bad parameters before committing to any significant action
- write your own routine to handle any cleanup necessary.

The former course is strongly recommended.

T-MSGX-KEY should be set to your own message for bad parameters, which might be patterned on the TB-5600 or TB-9000 messages in the distribution version. To implement an installation-wide standard for such messages, put the standard key in an installation-standard copybook and the corresponding message in an installation-standard message table. Many users will wish to add their own error exit routines, sending a message and then returning control to tablesONLINE/CICS.

Sample code follows:

```
*
 User-label.
*
*     SETUP MESSAGE - whatever
*
      MOVE message code              TO  T-MSGX-KEY.
```


This is the key required to find the message in the message table,

```
MOVE n                          TO  T-MSGX-INSERT1-LENGTH.
MOVE text                       TO  T-MSGX-INSERT1-VALUE.
```

This is the length and value for the first character string to overlay onto the message text (length zero if not used).

```
MOVE n                          TO  T-MSGX-INSERT2-LENGTH.
MOVE text                       TO  T-MSGX-INSERT2-VALUE.
```

This is the length and value for second character string, if any.

```
GO TO  Y100-NORMAL-EXIT.
```

This does not transfer to Y700-SEND-MESSAGE. In the typical case, normal restoration of the command area, library list, and abend status is desired, so the usual Y100-NORMAL-EXIT is used. At this point, the program is almost complete. There is no more user code to be run, and no cleanup to do, just a message to handle.

```
*
 Y700-SEND-MESSAGE.
*
*    FOR MESSAGE CALL MESSENGER PROGRAM
*
     EXEC CICS LINK         PROGRAM  (H-PROG-DKTBMSTK)
                            COMMAREA (T-MSGX-DFHCOMMAREA)
                            LENGTH   (H-DFHCOMM-MSGLENGTH)
                            END-EXEC.
*
```

This stacks the message in a buffer. tablesONLINE/CICS will retrieve it from the stack and display it on the screen at the appropriate time. The messages are not written to the screen directly. The user may not need the messages at this point nor wish to have the screen overwritten.

```
*
     IF  T-TBLX-BYPASS-ACTION-IND =  SPACES
         MOVE T-MSGX-TYPE          TO  T-TBLX-BYPASS-ACTION-IND.
*
```

If user code has already set the bypass indicator, it is left untouched. Otherwise, it is used to flag the type of message and related action:

- E—Error
- W—Warning
- I—Information
- A—Abend.

With the W and I types, tablesONLINE/CICS delivers the message but treats the operation as successful and carries on processing with the data as returned by the exit program. E indicates the operation failed; tablesONLINE/CICS will require that this error be dealt with by the terminal user.

After handling a message, the logic path falls through to RETURN-TO-CALLER. The exit program reaches this point either directly from the NORMAL-EXIT routine if there is no message, after any message set up by INVALID-CALL, or after a user routine has been dealt with.

User code should not normally branch directly to this label, since the tablesONLINE/CICS environment should be restored correctly first. Using NORMAL-EXIT is considerably safer.

```
*
 Y900-RETURN-TO-CALLER.
*
*    DURING TESTING PUT IN FOLLOWING CODE
*
*        MOVE user-debug-message       TO  T-MSGX-KEY
*        MOVE 8                        TO  T-MSGX-INSERT1-LENGTH
*        MOVE H-PROGRAM                TO  T-MSGX-INSERT1-VALUE
*        MOVE 12                       TO  T-MSGX-INSERT2-LENGTH
*        MOVE T-TBLX-PARM-INDICATORS   TO  T-MSGX-INSERT2-VALUE
*        EXEC CICS LINK       PROGRAM  (H-PROG-DKTBMSTK)
*                             COMMAREA (T-MSGX-DFHCOMMAREA)
*                             LENGTH   (H-DFHCOMM-MSGLENGTH)
*                             END-EXEC.
*
```

With this code inserted, every exit program invocation produces at least one message displaying the program name and the indicators with which it was called. This can be useful in testing and debugging exit programs.

The message itself can be created by copying and modifying TB-9001, the corresponding message for exit programs used internally by tablesONLINE/CICS.

```
*
 Y999-RETURN-TO-CALLER.
*
     EXEC CICS RETURN           END-EXEC.
     GOBACK.
*
**** END OF PROGRAM ********************************************
```


The EXITWS copybook

Here we describe the copybook EXITWS in which exit program mandatory fields are defined.

```
*****************************************************************
*                                                               *
*  WORKING-STORAGE - TABLESONLINE EXIT PROGRAM REQUIRED FIELDS  *
*                                                               *
*****************************************************************
*
 01  H-PROGRAM-CONTROL-AREA.
     05  H-COMMAND-AREA.
         10  H-COMMAND               PIC X(2).
         10  H-TABLE                 PIC X(8).
         10  H-FOUND                 PIC X(1).
         10  H-INDIRECT              PIC X(1).
         10  FILLER                  PIC X(2).
         10  H-ERROR                 PIC S9(4) COMP.
         10  H-COUNT                 PIC S9(9) COMP.
         10  H-LOCK                  PIC X(8).
* REL 5 OR LATER COMMAND EXTENSION
         10  H-ITEM-LENGTH           PIC S9(9) COMP.
         10  H-ACTUAL-ITEM-LENGTH    PIC S9(9) COMP.
         10  H-FG-KEY-LENGTH         PIC S9(4) COMP.
         10  H-FUNCTION-ID           PIC S9(4) COMP.
         10  H-FUNCTION-PARM.
             15  H-DATE              PIC 9(8).
             15  RESERVED            PIC X(20).
         10  H-RETURNEDABS-GEN       PIC S9(4) COMP.
         10  H-ERROR-SUBCODE         PIC S9(4) COMP.
```

A tableBASE command area for the exit program so that it can use tableBASE services without affecting the tablesONLINE/CICS command area.

```
     05  H-LIB-LIST                  PIC X(80).
     05  H-STATUS-SAVE               PIC X(80).
```

An area to store a copy of the library concatenation list so that if the exit program alters it, the original can be restored before returning to tablesONLINE/CICS.

```
     05  H-RETURN-CODES.
         10  H-NORMAL-RETURN         PIC X     VALUE ' '.
         10  H-BYPASS-RETURN         PIC X     VALUE 'Y'.
         10  H-INFO-RETURN           PIC X     VALUE 'I'.
         10  H-WARN-RETURN           PIC X     VALUE 'W'.
         10  H-ERROR-RETURN          PIC X     VALUE 'E'.
         10  H-ABORT-RETURN          PIC X     VALUE 'A'.
         10  FILLER                  PIC X     VALUE ' '.
     05  H-ABEND-STATUS-SWITCHES.
         10  H-ABEND-STATUS-OFF      PIC X(8)  VALUE 'NN      '.
         10  H-ABEND-STATUS-ON       PIC X(8)  VALUE 'YN      '.
```


Constants used in communicating with tablesONLINE/CICS.

```
05  H-CONSTANTS.
    10  H-PROG-DKTBMSTK         PIC X(8)  VALUE 'TBDKMSTK'.
    10  H-DFHCOMM-MSGLENGTH     PIC S9(4) COMP VALUE +148.
```

The program to call for message handling and the size of its parameter area.

```
    10  H-MSG-INVALID-CALL      PIC X(7)  VALUE 'TB-9000'.
    10  H-MSG-DEBUG-CALL        PIC X(7)  VALUE 'TB-9001'.
```

Areas where the copybook could be modified to provide for installation-wide standard messages similar to those with the above message keys. These messages could be created by modifying the distribution version messages TB-9000 and TB-9001.

The EXITPARM copybook

The next several pages describe the copybook EXITPARM which defines the tablesONLINE/CICS Transient Work Area. It should be included in all user exit programs.

Note: Areas labelled RESERVED or FILLER here are used internally by tablesONLINE/CICS or tableBASE and should not be tampered with.

```
*----------------------------------------------------------------*
* COBOL COPY BOOK MAPPING THE PARAMETERS PASSED TO TABLESONLINE
* CICS EXIT PROGRAMS VERSION 6 (RELEASE 5.1 FORMAT)
*----------------------------------------------------------------*
*                                                                *
     EJECT
******************************************************************
*                                                                *
 01  T-TWA.
*                                                                *
******************************************************************
*
*    TRANSACTION MANAGEMENT AREA POINTERS FOR TBLBASE
*
   03    T-TBPARM.
         10  RESERVED                  PIC X(64).
******************************************************************
*                                                                *
   03    T-SESSION-TABLE-ITEM.
*                                                                *
******************************************************************
     05  T-TERMINAL-ID                 PIC X(4).
     05  T-TERMINAL-INPUT.
         10 T-TERMINAL-APPL-ID          PIC X(04).
         10 T-TERMINAL-USER-ID          PIC X(08).
     05  T-SESSION-KEY.
         10 T-SESSION-TSQ-ID            PIC X(02).
         10 T-SESSION-ID.
            15 T-SESSION-PRIM-ALLOC     PIC S9(8) COMP.
```

```
              15 T-SESSION-WINDOW-ID       PIC S9(4) COMP.
```

These fields are used to uniquely identify the tablesONLINE/CICS session and window which invoked the exit.

```
************************************************************
*    PROGRAM IN PROGRESS CONTROLS
************************************************************
     05  T-PROGID-IN-PROCESS             PIC X(8).
     05  RESERVED                        PIC X(8).
```

This is fundamental information for tablesONLINE/CICS. User code should treat it as strictly read-only data.

```
****************************************************************
*    PARM AREA FOR HOOK PROCESSING    PARM-ID-TBL-ITM-FLD
****************************************************************

     05  T-TBLX-PARMS.
         10  T-TBLX-PARM-INDICATORS.
             15  T-TBLX-EXIT-INDICATORS.
                 20  T-TBLX-EXIT-STIF-IND    PIC X.
                 20  T-TBLX-EXIT-IO-IND      PIC X.
                 20  T-TBLX-EXIT-BA-IND      PIC X.
             15  T-TBLX-BYPASS-ACTION-IND    PIC X.
             15  T-TBLX-ALL-UPD-IND          PIC X.
             15  T-TBLX-KEY-UPD-IND          PIC X.
             15  T-TBLX-ITEM-UPD-IND         PIC X.
             15  T-OPER-PGM-OP-MODE          PIC X.
             15  T-TBLX-ITEM-ACTION          PIC X.
             15  T-TBLX-DUPSOK-IND           PIC X.
```

These are the main control variables for exit programs. They are described earlier in this chapter, in the section Interacting with tablesONLINE/CICS. The Exit Indicators and Bypass Action Indicator are also described in the comments in the exit program.


```
10  T-TBLX-PARM-VALUES.
    15  T-DSPL-CONV-FIELD-ERROR     PIC S9(4) COMP.
    15  T-FDTX-ERR-FIELD-COUNT      PIC S9(8) COMP.
    15  T-TBLX-CURR-ITM-COUNT       PIC S9(8) COMP.
```

These three fields are used for flagging field conversion errors in the tablesONLINE/CICS editor. The first identifies a field conversion error, the second the field to highlight in the multi-field item display, and the third identifies the row in the table. All three may be set by exit programs. For example, an IXF exit might set ERR-FIELD-COUNT so that tablesONLINE/CICS would scroll to the problematic field. Developers altering these must ensure that only sensible values are placed here. Count values pointing beyond end-of-row or outside the table can lead to erratic behavior.

```
    15  T-TBLX-TARGET-COUNT         PIC S9(8) COMP.
```

The tablesONLINE/CICS row move command uses the tableBASE Delete by Count (DC) and Insert by Count (IC) commands. (In multi-user update mode, DK and IK are used.) T-TBLX-TARGET-COUNT is the count the IC command requires to put the row in the right place. tablesONLINE/CICS has already allowed for such considerations as whether the target was flagged A (after) or B (before) and whether the deleted row was before or after the target.

```
    15  T-TBLX-PASSWD         PIC X(8).
    15  T-TBLX-GENERATION     PIC S9(8) COMP.
```

Specifies the password and generation number of the table for which tablesONLINE/CICS called the exit program.

```
    10  T-TBLX-ITEM-ENQUEUED            PIC X.
    10  RESERVED                        PIC X(5).
    10  T-OPER-TBLX-NAME                PIC X(8).
    10  RESERVED                        PIC X(6).
*
    10  T-SCRL-FIRST-SCR-CNT            PIC S9(8) COMP.
```

This field is used to set the count of the first row to appear on the Edit Table screen.


```
*****************************************************************
*    LINE COMMAND INFORMATION AREA
*****************************************************************

     05  T-LCMD-COUNT                    PIC S9(8) COMP.
     05  T-LCMD-ID                       PIC X.
     05  RESERVED                        PIC X(7).
* PLEASE NOTE THE OFFSET OF THE FOLLOWING SUBPARAMETER HAS
* INCREASED BY 4 BYTES FOR Version 6. ALL OTHER SUBPARAMETERS
* ARE IN THE IDENTICAL POSITIONS OF RELEASE 5.1.
     05  T-LCMD-NO-OF-ITEMS              PIC S9(8) COMP.
     05  RESERVED                        PIC X(24).
```

These fields are used in processing Line Commands on the Edit Table screen in a CSA exit. LCMD-COUNT contains the count of the first row of a Line Command as presented to the CSA exit. LCMD-ID identifies the Line Command. LCMD-NO-OF-ITEMS contains the range of items in a block line command as presented to the CSA exit. It has a value of zero if the command is a single character (non-block) command.

```
*****************************************************************
*     THIS IS THE DFHCOMM WORK AREA FOR MSGS PROCESSING
*****************************************************************

     05  T-MSGX-DFHCOMMAREA.
         10  RESERVED                    PIC X(8).
         10  T-MSGX-KEY                  PIC X(7).
         10  T-MSGX-FOUND                PIC X.
         10  T-MSGX-INSERT1-LENGTH       PIC S9(8) COMP.
         10  T-MSGX-INSERT1-VALUE        PIC X(60).
         10  T-MSGX-INSERT2-LENGTH       PIC S9(8) COMP.
         10  T-MSGX-INSERT2-VALUE        PIC X(60).
         10  T-MSGX-TYPE                 PIC X.
         10  RESERVED                    PIC X(3).
*
```

To send a message an exit program sets KEY and the various INSERT-xxxxx fields here.

When NORMAL-EXIT then calls the tablesONLINE/CICS message handler, that program takes a series of actions:

1. Looks up the KEY in the message table
2. Builds a message from the retrieved text and the two INSERT-VALUEs
3. Puts that message on a queue for the tablesONLINE/CICS screen handler
4. Reports success or failure of the above in FOUND
5. Puts the message type (A, E, I, or W) in TYPE here.

NORMAL-EXIT then copies TYPE to BYPASS-ACTION-INDicator, unless the user code has already set that indicator.


```
*****************************************************************
*    ITEM AREA FOR PF KEY/COMMAND TRANSLATION.
*****************************************************************

     05  T-PFKS-TABLE-ITEM.
         10  RESERVED                    PIC X(5).
         10  T-COMMAND                   PIC X(20).
         10  T-CMD-PARM                  PIC X(79).
     05  T-CMD-PARM-COUNT                PIC S9(8) COMP.
     05  T-CMD-COUNT-VALUE               PIC S9(8) COMP.
     05  RESERVED                        PIC X(132).
*
```

T-COMMAND holds the tablesONLINE/CICS command the user gave; T-CMD-PARM contains the command-line arguments to that command. T-CMD-PARM-COUNT is the length of the contents of T-CMD-PARM. T-CMD-COUNT-VALUE contains zero at the CSB exit; after this exit it contains either the numeric value entered into T-CMD-PARM, if any, or the relative cursor position. This is typically used with all commands which take a numeric parameter (DOWN, UP, FREEZEKEYS, etc.).

```
*****************************************************************
*    APPLICATIONS LIB LIST
*****************************************************************

     05  T-LIB-LIST.
         10  T-LIBDDN                    PIC X(8)
                                 OCCURS 10
                                 INDEXED BY T-LIB-LIST-INDEX.
*
```

The tableBASE library concatenation list which the calling tablesONLINE/CICS application is using in its searches. This area is read-only.


```
*****************************************************************
*    COMMAND AREA TABLE XXXXXXXXX          PARM-TBL-ITM-FLD
*****************************************************************
      05  T-TBLX-COMMAND-AREA.
          10  T-TBLX-COMMAND              PIC  X(02).
          10  T-TBLX-TABLE                PIC  X(08).
          10  T-TBLX-FOUND                PIC  X.
          10  T-TBLX-INDIRECT-OPEN        PIC  X.
          10  T-TBLX-RESERVED             PIC  X.
          10  T-TBLX-ABEND-OVERRIDE       PIC  X.
          10  T-TBLX-ERROR                PIC S9(04) COMP.
          10  T-TBLX-COUNT                PIC S9(08) COMP.
          10  T-TBLX-LOCK-LATCH           PIC  X(08).
*** REL 5 OR LATER COMMAND EXTENSION
          10  T-TBLX-ITEM-LENGTH          PIC S9(08) COMP.
          10  T-TBLX-ACTUAL-ITEM-LENGTH   PIC S9(08) COMP.
          10  T-TBLX-FG-KEY-LENGTH        PIC S9(04) COMP.
          10  T-TBLX-FUNCTION-ID          PIC S9(04) COMP.
          10  T-TBLX-FUNCTION-PARM.
              15  T-TBLX-DATE             PIC  9(08).
              15  RESERVED                PIC  X(20).
          10  T-TBLX-ABS-GEN-NO           PIC S9(04) COMP.
          10  T-TBLX-ERROR-SUBCODE        PIC S9(04) COMP.
```

The tableBASE command area for tablesONLINE/CICS is read-only in most applications. The exception occurs when the exit program replaces the tableBASE action with its own. Consider an exit program that operates at item input time replacing tableBASE retrieval operations with versions that only retrieve certain types of rows. This program must update T-TBLX-COUNT to point to the row actually retrieved.

Developers of such programs must ensure that only valid values are written here; tablesONLINE/CICS does no error checking on this area since normally only tableBASE writes here and tableBASE will never report, for example, a count that points beyond the end of a table.


```
*****************************************************************
*    DEFINITION AREA FOR TABLE xxxxxxxx
*****************************************************************

     05  T-TBLX-DEF-AREA.
         10  T-TBLX-DEF-O-M-T-S.
             15  T-TBLX-DEF-ORG          PIC X.
             15  T-TBLX-DEF-MTHD         PIC X.
             15  T-TBLX-DEF-TYPE         PIC X.
             15  T-TBLX-DEF-SMC          PIC X.
         10  T-TBLX-DEF-RPSWD            PIC X(8).
         10  T-TBLX-DEF-WPSWD            PIC X(8).
         10  T-TBLX-DEF-ISZ              PIC S9(8) COMP.
         10  T-TBLX-DEF-KSZ              PIC S9(8) COMP.
         10  T-TBLX-DEF-KLOC             PIC S9(8) COMP.
         10  T-TBLX-DEF-EST              PIC S9(8) COMP.
         10  T-TBLX-DEF-GENS             PIC S9(4) COMP.
         10  T-TBLX-DEF-EXP              PIC S9(4) COMP.
         10  T-TBLX-DEF-LO-DEN           PIC S9(4) COMP.
         10  T-TBLX-DEF-HI-DEN           PIC S9(4) COMP.
         10  RESERVED                    PIC X(6).
         10  T-TBLX-DEF-DATE-TIME-2000.
             15  T-TBLX-DEF-CENTURY      PIC X(2).
             15  T-TBLX-DEF-DATE-TIME    PIC X(10).
         10  T-TBLX-DEF-ABS-GEN          PIC S9(4) COMP.
****  THESE FIELDS USED ONLY BY 'GD' COMMAND.
         10  T-TBLX-DEF-DSN              PIC X(44).
         10  T-TBLX-DEF-REL-GEN          PIC S9(4) COMP.
         10  T-TBLX-DEF-GEN-CNT          PIC S9(4) COMP.
         10  T-TBLX-DEF-MAX-ITEMS        PIC S9(8) COMP.
         10  T-TBLX-DEF-DDNAME           PIC X(8).
         10  T-TBLX-DEF-DATA-TABLE       PIC X(8).
         10  T-TBLX-DEF-OPEN-STATUS      PIC X.
         10  T-TBLX-DEF-ALTS-INVOKED     PIC X.
****  THESE FIELDS USED FOR 5.0 UPGRADE.
         10  T-TBLX-DEF-VERSION          PIC X.
         10  RESERVED                    PIC X.
         10  T-TBLX-DEF-USER-ID          PIC X(8).
         10  T-TBLX-DEF-VIEW-DATA-NAME   PIC X(8).
         10  RESERVED                    PIC X(12).
         10  T-TBLX-DEF-USER-COMMENTS    PIC X(16).
         10  RESERVED                    PIC X(76).
*
```

This is the tableBASE DT block, the table definition stored in the tableBASE library for the table for which tablesONLINE/CICS called the exit program. This area is read-only.

See the documentation for the DT command in Chapter 3 and the DEFINITION-BLOCK parameter in Chapter 4, or the table definition section of the *tablesONLINE/CICS User's Guide* to interpret these fields.


```
************************************************************
*    TRAILER AREA OF VIEW TABLE             PARM-TBL-ITM-FLD
************************************************************

     05  T-TRLX-TABLE-ITEM.
         10  T-TRLX-DSPL-SEQ             PIC 9(5).
         10  T-TRLX-TBLDEFN-SUFFIX       PIC X.
         10  T-TRLX-KEY-IND              PIC X.
         10  T-TRLX-FLD-NAME             PIC X(20).
         10  RESERVED                    PIC X(8).
         10  T-TRLX-HELP-TABLE           PIC X(8).
         10  T-TRLX-EXITPGM-VERSION      PIC X.
         10  RESERVED                    PIC X(8).
         10  T-TRLX-TBLX-SUFFIX-LNGTH    PIC S9(4) COMP.
         10  T-TRLX-TBLX-SUFFIX-LOCN     PIC S9(4) COMP.
         10  T-TRLX-TBLX-DUPSOK-IND      PIC X.
         10  T-TRLX-TBLX-KEY-PROT-IND    PIC X.
         10  T-TRLX-TBLX-REORG-CODE      PIC X.
         10  T-TRLX-TBLX-MULTI-USER-IND  PIC X.
         10  T-TRLX-ITEM-HK-PGM          PIC X(8).
         10  T-TRLX-ITEM-HK-IN-IND       PIC X.
         10  T-TRLX-ITEM-HK-OUT-IND      PIC X.
         10  T-TRLX-ITEM-HK-XFLDCHK-IND  PIC X.
         10  T-TRLX-ITEM-EXCLUDE-IND     PIC X.
         10  T-TRLX-TABL-HK-PGM          PIC X(8).
         10  T-TRLX-TABL-HK-IN-IND       PIC X.
         10  T-TRLX-TABL-HK-OUT-IND      PIC X.
         10  T-TRLX-TABL-HK-CLOSE-IND    PIC X.
         10  T-TRLX-TABL-HK-UTIL-IND     PIC X.
         10  T-TRLX-TABL-HK-DEF-NEW-IND  PIC X.
         10  T-TRLX-TABL-HK-DEF-UPD-IND  PIC X.
         10  T-TRLX-TABL-HK-DEF-GET-IND  PIC X.
         10  RESERVED                    PIC X.
         10  T-TRLX-TBLX-ISZ             PIC S9(4) COMP.
         10  T-TRLX-TBLX-KLOC            PIC S9(4) COMP.
         10  T-TRLX-TBLX-KSZ             PIC S9(4) COMP.
         10  T-TRLX-FREEZE-KEY-CNT       PIC S9(4) COMP.
         10  T-TRLX-ROLL-THRU-FDT-IND    PIC X.
         10  RESERVED                    PIC X(103).
*
```

This is the View trailer area. Many of these fields are documented in the section on Supplementary View Information in the Defining Tables chapter of the *tablesONLINE/ CICS User's Guide*, which describes editing this trailer.

The three fields here, T-TRLX-TBLX-ISZ, T-TRLX-TBLX-KLOC, and T-TRLX-TBLX-KSZ, are respectively, the ITEM (ROW) SIZE, KEY LOCATION, and KEY SIZE as calculated from the View by tablesONLINE/CICS. These are checked at Table Open time for consistency with the tableBASE library information by tablesONLINE/CICS internal code, and the open fails if an inconsistency is encountered.


```
*****************************************************************
*    COMMAND AREA FOR VIEW TABLE              PARM-ITM-FLD
*****************************************************************
 05  T-FDTX-COMMAND-AREA.
         10  T-FDTX-COMMAND              PIC X(2).
         10  T-FDTX-TABLE                PIC X(8).
         10  T-FDTX-FOUND                PIC X.
         10  T-FDTX-INDIRECT-OPEN        PIC X.
         10  T-FDTX-FILLER               PIC X.
         10  T-FDTX-ABEND-STATUS         PIC X.
         10  T-FDTX-ERROR                PIC S9(4) COMP.
         10  T-FDTX-COUNT                PIC S9(8) COMP.
         10  T-FDTX-LOCK-LATCH           PIC X(8).
* REL 5 OR LATER COMMAND EXTENSION
         10  T-FDTX-ROW-OVERRIDE-LENGTH  PIC S9(8) COMP.
         10  T-FDTX-ROW-ACTUAL-LENGTH    PIC S9(8) COMP.
         10  T-FDTX-FG-KEY-LENGTH        PIC S9(4) COMP.
         10  T-FDTX-FUNCTION-ID          PIC S9(4) COMP.
         10  T-FDTX-FUNCTION-AREA.
             15  T-FDTX-DATE             PIC 9(8).
         10  T-FDTX-RESERVED             PIC X(24).
```

This is the tableBASE command area tablesONLINE/CICS uses to retrieve View information for the table it is operating on. This area is read-only.

```
*****************************************************************
*    ITEM AREA OF VIEW TABLE                  PARM-FLD
*****************************************************************

     05  T-FDTX-TABLE-ITEM.
         10  T-FDTX-DSPL-SEQ             PIC 9(5).
         10  T-FDTX-DSPL-KEY       REDEFINES T-FDTX-DSPL-SEQ.
             15  T-FDTX-99KEY            PIC XX.
             15  T-FDTX-999SUFFIX        PIC XXX.
         10  T-FDTX-TBLDEFN-SUFFIX       PIC X.
         10  T-FDTX-KEY-IND              PIC X.
         10  T-FDTX-FLD-NAME             PIC X(20).
         10  T-FDTX-XTND-IND             PIC X.
         10  T-FDTX-FLDN-EXTATT          PIC X(4).
         10  T-FDTX-DSPL-LNGTH           PIC S9(4) COMP.
         10  T-FDTX-DSPL-FORMT           PIC X.
         10  T-FDTX-DSPL-ATTR            PIC X.
         10  T-FDTX-DSPL-FEAT            PIC X.
         10  T-FDTX-DSPL-EXTATT          PIC X(4).
         10  RESERVED                    PIC X(10).
         10  T-FDTX-EXITPGM-VERSION      PIC X.
         10  T-FDTX-TBLX-LOCN            PIC S9(8) COMP.
         10  T-FDTX-TBLX-LNGTH           PIC S9(8) COMP.
         10  T-FDTX-TBLX-FORMT           PIC X.
         10  T-FDTX-EDIT-ACTION          PIC X.
         10  T-FDTX-EDIT-TBLNAME         PIC X(8).
         10  T-FDTX-EDIT-INPT-REF        PIC X(20).
```



```
           10  T-FDTX-EDIT-HK-PGM          PIC X(8).
           10  T-FDTX-EDIT-HK-IN-IND       PIC X.
           10  T-FDTX-EDIT-HK-OUT-IND      PIC X.
           10  RESERVED                    PIC X(100).
*
*****************************************************************
```

This area holds the View row last retrieved by tablesONLINE/CICS. For field-level exits, it can be relied on to hold the View description of the current field.

Higher-level exits (item- and table-level) should not rely on information here. There is no way to know in an item-level or table-level exit what field was last accessed.

```
*****************************************************************
*    EXIT USER AREA IN TRANSACTION WORK AREA  (96 BYTES)
*****************************************************************

     05  T-EXIT-USER-AREA               PIC X(96).
```

This area is reserved for exit-to-exit data passing and is entirely controlled by the exit developer.

CAUTION: Some caution is necessary if several exits in one application use this area since there is only one such area. It is possible to create a situation in which exit program A leaves data here for exit B but exit X overwrites it before B sees it. If B blindly accepts it as valid data, this can become messy.

Where there is a potential problem the following recommendation applies: use part of the area to identify the writing process so that the reading process can validate at least that the data available was intended for it.


```
*****************************************************************
*    ITEM AREA FROM PROFILE TABLE
*****************************************************************

     05  RESERVED                        PIC X(1396).

*****************************************************************
*    ITEM AREA FOR MENU OPTION TABLE
*****************************************************************

     05  T-MENU-TABLE-ITEM.
         10  T-MENU-TRANSFER-OPTION.
             15  T-MENU-MAP-ID           PIC X(8).
             15  T-MENU-SELECT-SYMBOL    PIC X(4).
         10  T-MENU-TBLDEFN-SUFFIX       PIC X.
         10  T-MENU-SELECT-SYMBOL-X      PIC X(4).
         10  T-MENU-DESC-SHORT           PIC X(20).
         10  T-MENU-DESC-SHORT-X         PIC X(4).
         10  T-MENU-DESC-LONG            PIC X(50).
         10  T-MENU-DESC-LONG-X          PIC X(4).
         10  T-MENU-DISPLY-SW            PIC X(1).
         10  T-MENU-TRANSFER-PGM         PIC X(8).
         10  T-MENU-PGM-OP-MODE          PIC X.
         10  T-MENU-TBOL-SYSTEM-PGM      PIC X.
         10  T-MENU-TRANSFER-TRANID      PIC X(4).
         10  T-MENU-NEXT-SELECT          PIC X(4).
         10  T-MENU-TRANSFER-PARMS.
           12  T-MENU-TRANSFER-PARM1A.
            15  T-TRANS-MENU-MAP-ID      PIC X(8).
            15  T-TRANS-MENU-SEL-SYMBOL  PIC X(4).
            15  RESERVED                 PIC X(54).
           12  T-MENU-TRANSFER-PARM1B  REDEFINES
                                       T-MENU-TRANSFER-PARM1A.
            15  T-MENU-DBTYPE            PIC X(8).
            15  T-MENU-TBLX-NAME         PIC X(8).
            15  T-MENU-TBLX-FUNCTION     PIC X(8).
            15  T-MENU-TABLE-LIB         PIC X(8).
            15  T-MENU-SPECIAL-FDTX      PIC X(8).
            15  T-MENU-TBLX-RECOVER-IND  PIC X.
            15  T-MENU-TBLX-LOG-IND      PIC X.
            15  T-MENU-TBLX-DUPSOK-IND   PIC X.
            15  T-MENU-UTIL-FUNCTION-KEY PIC X(8).
            15  T-MENU-VIEW-LIB          PIC X(8).
            15  RESERVED                 PIC X(7).
```

This area contains the row from the menu table corresponding to the selected menu option for Menu Command-level exits CMB and CMA.


```
*
  03  RESERVED                          PIC X(2352).
************************************************************
*    TBPARM FOR EDITING/BROWSING TABLE IN REMOTE VTS
*
************************************************************
   03  T-TRPARM.
         10  RESERVED                    PIC X(64).
*
****************************************************************
*  END OF USER ACCESSIBLE EXITPARM
****************************************************************
```

This last area is the new TBPARM created for Version 6 and is required when a table is opened in a VTS-TSR that is different from the designated systems VTS-TSR. If your installation has opted for VTS, and if you are writing exits using a table in a VTS-TSR that is not the systems VTS-TSR, then you should review the exit programs currently in use and determine if the exit is processing the same table that is being operated in by tablesONLINE/CICS. If it is, change the tableBASE calls in the exit program to use T-TRPARM rather than T-TBPARM. If you are unsure whether VTS is in use, making this change will not affect the operation of existing exits not using VTS-TSR processing, as TRPARM defaults to TBPARM if the table is in a local TSR.

Exit program scenarios

The rest of this chapter discusses how exit programs may be used to achieve various objectives in designing an application. It consists mainly of text with samples of code.

The examples vary considerably in complexity; some are intended to show the basic code required for a given exit task, while others point out subtleties or explore more difficult applications. Readers are encouraged to be selective in reading this section. There is no requirement to read this section sequentially.

The following convention is used throughout the section: TOB exit program refers to "an exit program called with the indicators TOB".

Security

tablesONLINE/CICS uses the security mechanisms of its environment — CICS login and transaction-access security and the tableBASE password mechanism — so some security is achieved without writing exit programs.

Where tighter security is required, you can make the tablesONLINE/CICS System sign-on exit invoke a security package such as RACF, ACF2, or Top Secret. All user and application IDs passed to tablesONLINE/CICS are then validated by the security package. Typically, there is only one such exit program per installation and developers do not have access to it, so it will not be covered in any detail here. See the security section in the *tableBASE Administration Guide*.

User exits could also be written for other security tasks. Consider the table with critical data that should be accessible only to certain employees. This might be protected by making it accessible only to one application and limiting access to that application via the Application Control Table (ACT), with or without a System sign-on exit call to a security package. The tableBASE password mechanism would also protect the table. These are the usual protection methods.

There is also a third option. Install a TOB exit in that table's View to check a table of authorized users and refuse to open the table for others. This controls all tablesONLINE/CICS access and offers one advantage over using the ACT as a control mechanism—a manager could be given control over this mechanism without being given access to the ACT. In some applications, this ability to give departmental managers control over some security mechanisms for their own employees without compromising overall system security by allowing many people access to its mechanisms, might be essential.

Data encryption on non-key fields could be implemented either in field-level exit programs or in an IXF exit. The exit programs and Views involved would look much like those described below for the example which translates between state ZIP codes (NY, MI, etc.) and state names (New York, Michigan, etc.) except that they would be encrypting and decrypting with some user algorithm rather than simply encoding and decoding via tableBASE lookup.

Encryption of key fields would affect tableBASE searches so whole row data encryption, while possible, would be trickier. Hash searching on the encrypted keys would work provided the encryption algorithm guaranteed a unique encrypted form for each unique key. You would have to decrypt in the IIB exit and encrypt in the item-level output exits.

Note: DataKinetics recommends that you do not write encryption programs. Writing efficient and effective encryption code is by no means an easy task and it would be pointless except in a system where all aspects of security have been subjected to thorough analysis and other precautions already taken.

Making rows invisible to some users

A concern which straddles the border of security and application management is the provision of item (row)-level control over data access. Other mechanisms provide application-level and table-level control, but for finer work, exit programs are needed.

In a personnel application, for example, you might wish to ensure that only selected staff could access the rows describing corporate top management.

This can be done in two ways. You could write an IIA exit to check each row and return only those the user is allowed to see. The sample exit program demonstrates this method. Alternatively, you could make accesses to the Data Table indirect and write a TOB exit to create an Index table for each user. All checking would then be done when the table was opened rather than at row input time. Which method is appropriate depends mainly on how many rows will be accessed per session. For small numbers, check each row; for many, build an Index.

Another example in this area might be to prevent any employee from updating his or her own record. Either of the exclusion methods just described could easily be modified to do this and would entirely control access to the record.

This test could also be done in an IXF exit. Unlike other implementations, this allows the employee to see his or her own record. The IXF exit is called for every output action, so using it avoids such problems as writing a trap for update actions and having someone avoid the trap by deleting the old row and creating a new one.

Interfacing

Some applications need to use data not stored in tableBASE tables—perhaps the data is stored in a VSAM file or a database. Conversion exits operating at either table or row level can support editing non-table data with tablesONLINE/CICS. This can be done with either table-level or row-level exits.

At the table level, you can write a TOB exit to load the data into a tableBASE table and a TSB exit to move it back. Both should set the BYPASS indicator to Y to indicate that they have taken care of the action themselves and tablesONLINE/CICS should bypass it. A TCB or TCA exit may not be needed, since tablesONLINE /CICS can delete a table from memory without the assistance of an exit program. One of these exits might be used for housekeeping functions, as is done in the example on the next few pages.

Performing the same function at the item (row) level is more difficult, since there are more cases to be handled—new row, update, delete, move — and some cases may not map well onto the external data representation. It will also typically involve higher overheads, since every operation requires access to the external data. It is possible and might be necessary if, for example, there were other applications using the data and having tablesONLINE/CICS working on a private copy was not acceptable. Again, all the exit programs should be invoked before the action and should set the BYPASS indicator.

Interfacing to other processes as well as to external data is also possible through exit programs. You might, for example, have a TSA exit append the name of the stored table to a list which some batch backup process would use.

Process interfacing often involves constructing a transaction table to record changes on the main Data Table. Consider an application where, when an employee is added to the employee table, one or more messages need to be sent (for example, notes to his or her supervisor, to the systems person who will need to set up an account for him or her, and to the editor of the company newsletter). An incorrect method of implementing this would be to have an INA exit send the messages, because this can result in the messages being sent at the wrong time. If a user creates a row in the in-memory copy of the table, then exits with the <PF12> key so that the table is not stored to disk, it is too late to call back a message already sent.

One way to code this is with a TOA exit that creates a dynamic table for messaging information, an INA exit that adds a row to that table whenever a new row is added to the employee table, a TSA exit that sends messages and clears the associated rows from the dynamic table after the employee table has been stored, and a TCA exit that closes the dynamic table when the employee table is closed. The dynamic table records transactions against the employee table, and its records drive the messaging application.

This is an extremely powerful and flexible means of interconnecting programs. With carefully designed exit programs and table structures, it is possible to use the tablesONLINE/CICS table editor as the front end of data-driven systems of whatever level of complexity an application requires. Example code follows for the central part of an exit program that, for specified table names, provides an interface to external data.

```
*
*    TEST FOR SPECIFIC TABLES
*
     IF   T-TBLX-TABLE  =  'table1 '
          GO TO  table1-CONTROLLER.
     IF   T-TBLX-TABLE  =  'table2 '
          GO TO  table2-CONTROLLER.
*
*    ERROR IN CALLING EXIT PROGRAM - INVALID TABLE
*
     MOVE your key       TO  T-MSGX-KEY.
     MOVE 8              TO  T-MSGX-INSERT1-LENGTH.
     MOVE T-TBLX-TABLE   TO  T-MSGX-INSERT1-VALUE.
     MOVE ZERO           TO  T-MSGX-INSERT2-LENGTH.
     GO TO  Y100-NORMAL-EXIT.
*
****************************************************************
```

This code either branches to a label associated with a particular table or, if called for an unrecognized table name, sets up a message and branches to the exit routine that will handle the message and return control to tablesONLINE/CICS.

The NORMAL-EXIT routine will search the message table (normally TBOLMSGS, but this name may be tailored via the TBOLACT entries for the application) using the key stored at T-MSGX-KEY. A successful search will give it message TEXT, OFFSET information, and a single-character message TYPE, that in this case should be **E**. It then builds the actual message by inserting INSERT-VALUE text into the retrieved message TEXT according to the OFFSET and INSERT-LENGTH information it has, puts the results on a queue, and puts message TYPE into BYPASS-ACTION-IND.

On regaining control, tablesONLINE/CICS sees a BYPASS-ACTION-IND of **E** which causes it to treat the action as failed and return to the same screen. It also finds the message on the queue and delivers it.

This branching structure can accommodate any number of table names, and can either give each table its own label and code or send several branches to the same destination if several tables need similar handling.

An example of code that might appear at one of the target labels follows. This code, continued from above, handles interfaces to an external non-table data storage medium by moving data to a table at Table Open time and sending it out somewhere at Table Close time.

```
*
 table-CONTROLLER.
*
    IF  T-TBLX-EXIT-INDICATORS  = 'TOB'
        GO TO  TOB-CREATE-DYNAMIC-TABLE
    ELSE
        IF  T-TBLX-EXIT-INDICATORS  = 'TSB'
            GO TO  TSB-SUBMIT-JOB
        ELSE
            IF  T-TBLX-EXIT-INDICATORS  = 'TCA'
                GO TO  TCA-DELETE-WORK-TABLES
           ELSE
                GO TO  Y200-INVALID-CALL.
*
```

This code tests the three indicator bytes looking for valid combinations. The cases it handles are listed in Table 13-4.

Table 13-4: Exit indicators controlling multiple branch logic

Indicators	Called at level	Operation	Timing
TOB	Table	Open	Before
TSB	Table	Store	Before
TCA	Table	Close	After

On any other combination it branches to the INVALID CALL code. Any item-level or field-level exits required for this table must be implemented in separate exit code.

```
*
 TOB-CREATE-DYNAMIC-TABLE.
     user code to create a table in memory and fetch non-table data
     on success, set BYPASS-IND to 'Y'
     on failure, set BYPASS-IND to 'E' and set up message
     GO TO  Y100-NORMAL-EXIT.
 TSB-SUBMIT-JOB.
    user code to store table data to external medium
    (and/or submit a job based on it)
    on success, set BYPASS-IND to 'Y'
    on failure, set BYPASS-IND to 'E' and set up message
    GO TO  Y100-NORMAL-EXIT.
 TCA-DELETE-WORK-TABLES.
    user code for housekeeping functions
    tablesONLINE has already closed the in-memory table
    GO TO  Y100-NORMAL-EXIT.
 *
```

The three labels have the code beneath them to do the actual work of this exit program.

Field-level data validation

The most common application of exit programs is data validation. Some validation—checking field values against View data definitions — is built into tablesONLINE/CICS and does not require an exit program. This will, for example, ensure that only numeric values are entered into the salary field of a personnel record, or that the Date Of Hiring field holds a syntactically valid date string. Similarly, range checking or existence checking for field data is available through built-in View capabilities.

The built-in validation checks only test syntax. If not all numbers, dates, character strings or whatever “make sense” in your application, then field-level exits may be needed.

A straightforward example is an exit to test whether a field describing a customer discount code holds a sensible value if that code is only valid for specific time periods. This could be an FOB exit looking for mixed-case input values or an FOA exit checking for upper-case codes only, after tablesONLINE/CICS had moved the data and performed lower-case to upper-case translation.

Doing the test after the move often means a simpler test, as in this example. It has the disadvantage that bad data is not caught until after it has been transferred. In a field-level or item-level exit this cannot corrupt data; the bad data has only been moved in memory, not stored to disk, and if the exit program has set the BYPASS-ACTION-IND to **E**, processing will not proceed until the error is corrected, so no permanent damage can be done.

Some inconvenience can be created, since old data is overwritten by the transfer. In the example, when the user gets an error message about the customer discount field there is

only one way to discover what the value was before the error was made—cancel out of the row edit screen and reselect the row to pick up a fresh copy of the version on the table.

The sample exit EXITPGMC in the supplied education dataset performs a similar test for a gender field, with a few features thrown in to make a more interesting example.

Field-level exits can also perform specialized range checks and check for valid formats in entries such as part numbers, serial numbers, model numbers, invoice numbers, employee numbers, department codes, or any other data with a fixed format.

In many organizations it makes sense to provide a shared central library for reuse of tested exit code, thereby implementing corporate standards for important fixed format data.

Misused field exits

Where any of several fixed formats may apply to a single field, a field-level exit is not the solution. Canadian and British postal codes, for example, each have their own format, both including alphabetic characters that would be invalid in a US ZIP code. Clearly an exit program that merely looks at field data and accepts “any of the above” will miss some errors it should catch.

A field-level exit program controlled by information from other fields is not an acceptable solution to such problems, or indeed to any problem. For example, if an employee is being transferred from New York to London, then the value in the country field when the postal/ZIP exit is called depends on whether the user has updated that field yet. When writing the program, the exit program developer cannot know what order the user will later choose, or even in what order the fields will be presented, since this is user-controlled via Display Order editing in the View. Developers, then, must not rely on data in other fields to control field-level validation.

The general principle is that it is not wise to write a program whose behavior depends on something outside both its program code and the data it was intended to deal with.

A field-level exit program can safely update one field from data in another, for example, converting "MI" in a US state code to "Michigan" in a state name field, provided that:

- either the target field is protected from user data entry, or
- the conversion works both ways

so that the program code guarantees that inconsistent data cannot end up in the two fields.

A field-level exit cannot use data from other fields to control decisions, e.g., to determine which validation it should apply to its target field. At field validation time, all data from other fields must be considered unreliable since the developer cannot know its state. Any code controlled by this would itself be unreliable.

Cross-field validation

The solution to the problem described above is to delay postal/ZIP code checks until IXF exit time. At that point, all field changes should have been completed and it is perfectly sensible to check whether the postal/ZIP code field contains data valid for the country, as specified by the country field. Any error message generated should mention both fields so that the user is not left wondering why the system is complaining about a field containing perfectly good data.

In fact, at this point, any group of related fields can be checked against each other, against a table of acceptable combinations, or whatever. For example, the postal/ZIP code, country, city, state or province, and phone number might all be examined in an address validation exit.

The difficulty for the developer is deciding what should be checked and when. Deciding when is straightforward. To avoid unpredictable combinations of interdependent field data, some things can only be handled at IXF exit time. Any validation possible at the field level may also be delayed until IXF time if the developer finds this convenient. The difference is that, if set up as an FOB or FOA exit, the code will run every time the field is changed. However, if included in the IXF exit, it will run whenever any field of the row has changed and some output action is requested or ENTER has been pressed.

Deciding what to validate is more difficult. On the one hand, you want to validate as thoroughly as possible in order to catch errors before they have consequences, and to make the system's output as reliable as possible. On the other hand, the development effort and execution overhead for validation routines must be factored into the equation. When in doubt, lean toward thorough validation, particularly at IXF time, provided that validation rules are clearly specified.

Implementing a comprehensive address validation exit, for example, is tempting. Address information with essentially the same set of fields turns up in many different areas which could be subject to validation.

Unfortunately, with real world data, the rules are not always clear. Someone with a New Jersey phone number might get mail via an address at the company's New York headquarters, so just checking ZIP code versus area code doesn't always work.

Salary range checking

Another example of IXF validation is range checking on salaries in a personnel table. tablesONLINE/CICS will ensure that the salary field has a numeric value, but that is hardly an adequate constraint. Range validation capabilities are provided with field definitions in tablesONLINE/CICS Views, but this may not be enough for a thorough validation check.

A field-level exit might be written for salary fields, but what could it use for rules? Any test for a positive integer less than, say, 200,000 would be reasonable in most companies, but not adequate protection in any.

Implementing an inadequate rule is dangerous. Consider the less than 200,000 rule. A typo producing '92,000' instead of '29,000', for example, is accepted. The cheque ends up framed on the recipient's office wall.

A better rule would validate pay rates against limits for the job category. This requires access to category information from another field of the item, so it must wait until IXF exit time. Trying to do it at field validation time would create unreliable code as described under 'Misused Field Exits' above.

The code to implement this rule is just a table lookup with job category as key and an upper and lower boundary as retrieved data. Now the range check is more useful. Moreover, the system is easily maintained. If pay scales change, update the boundary table entries for the categories affected. All future validations will use the new scales.

Since all fields in the row are available at this point, more complex conditions could also be used. In some companies pay scales vary with location as well as by category. The New York office pays New York's "going rates"; Cleveland uses another scale. The plant in Left Overshoe, Manitoba offers higher salaries as "isolation pay" while the one in Trendville, CA offers even more so that staff can afford housing there.

This can readily be handled in any of several ways. One way is to put the complexity into the table, i.e., make location as well as category part of the key for looking up range boundaries. You could also handle the problem in the code by adding conditional branches in the exit program. Consider the difficulty of adding a new region in each case: it is far preferable to be able to do this by table editing than to have to modify and recompile code.

Timing is critical

Frequently, you want to maintain some form of consistency or synchronization between rows on a table, between rows on different tables, or between data and various system actions. The main question for the application designer is when each action needs to be performed to maintain the synchronization. For tablesONLINE/CICS, this often becomes which exit opportunity to use for an operation.

Any item-level or field-level exit program can, of course, look at any tables which the application does not change and may write to tables designed for a group of related exits, as described for previous examples under 'Interfacing'.

Consider the item-level exits in the employee table discussed in the previous example. The IXF exit can, as suggested above, read information from a salary range table. Other item-level exits might create rows in a message table when an employee row was added, or deleted, or when certain fields, such as salary, were updated. It would be a serious error to have item-level exits actually send such messages, since at the time of item-level operations you don't know that the user will actually follow through and commit the changed table to storage. Instead, you should accumulate message data from item-level exits and wait until the TSA exit to actually send it. At that point, you know the table has been successfully stored.

The general rule here is that the exit programs should not commit themselves to irrevocable actions until the user commits to storing the table.

Journaling

Where the concern is transaction logging for crash recovery, it is natural to do the logging at the item level and to treat Table Store actions as checkpoints.

In this case, each row output action (Move, Delete, New, or Update) should create a log entry with sufficient information to reconstruct the action if necessary. Table Store actions should also be marked in the journal.

Recovery then consists of searching backward in the journal to the last Table Store transaction, opening the table as it was last stored, and then moving forward through the journal, applying the item-level changes required, and storing the updated version.

If CICS temporary storage is used for journaling, then the journal will survive a warm restart of CICS but not a cold restart.

Complex synchronization

Synchronization issues may become quite complex, even within a single table. Consider comparing an employee's salary to that of the supervisor. Whether and when to do this depends on how significant the comparison is, which in turn depends greatly on the organization.

In some organizations, an employee salary higher than the manager's might be a routine occurrence and not worth checking at all. In others, it might be an oddity worth catching in an IXF exit and reporting to the operator as an information message. It is safe to use this exit when no more drastic action is taken.

In some organizations, such a row would be conspicuously anomalous and a more serious action such as rejecting the row or notifying the supervisor's supervisor would be appropriate. An item-level exit must not take serious actions based on other rows in the table. Just as cross-field validation must wait for IXF time, so serious cross-row checks must wait for TSB or TSA time. The required data is not reliable until then.

For example, if the IXF exit rejects a salary greater than the supervisor's, then the operator who attempts to update both salaries and takes the rows in the wrong order is put to some unnecessary trouble. The solution is to delay this check until the TSB exit.

Any other actions intended to ensure consistency within a table should also be taken at TSB exit time. For example, if the supervisor's row has a Number of Subordinates field, then this is the correct time to adjust it to reflect any change which the employee row change implies.

Actions which keep the outside world posted on the state of the table should, in general, be done at TSA exit time. For example, if the required action when an employee is paid more than a supervisor is to notify the supervisor's supervisor, then TSA is the correct time for such messages.

System synchronization

The most complex synchronization and data integrity issues arise when several data structures must be kept mutually consistent while in active use by several applications.

This document cannot deal adequately with these issues, though the techniques mentioned above for less complex problems may be of some use. A few principles are noted here.

The distinction between actions on items (rows) and actions on sets of rows (on tables within tableBASE or datasets elsewhere) is critical. In general, you want to keep these levels quite distinct.

For example, an item-level exit program will often:

- access a row from another table
- create a row on an intermediate Data Table
- create a row on a messages table, or
- create a row in a journal

and might reasonably:

- update rows in several tables
  (if related data were distributed so as to require that)

but would not normally:

- cause any table-level action, or
- take any other irrevocable action.

Irrevocable actions are better left until Table Store time.

In general, what will be needed is two systems of synchronization, one for item-level actions and another at the table level, with careful analysis of the relation between them and careful planning of what actions

Appendix A

Hints and suggestions

This appendix contains information you will find useful to optimize tableBASE use. Where applicable, the content is divided into advice that applies to tableBASE Version 6 and Version 5 respectively. Due to the numerous changes to Version 6, advice applicable to Version 5 is not necessarily appropriate to Version 6.

tableBASE space requirements

VSAM tableBASE libraries

For VSAM libraries, the default blocksize of 3120 (RECSZ in IDCAMS DEFINE) results in a CISZ of 3584, 13 blocks per track and 71.6% utilization. Increasing the RECSZ to 3577 (the maximum in a 3584 CISZ) results in a 82.1% utilization. Increasing it to 4091 (the maximum in a 4096 CISZ) results in 86.6% utilization. Reducing it to 2041 (2048 CISZ) results in 75.6% utilization; reducing it to 1017 (1024 CISZ) yields 59.2% utilization.

BDAM/QSAM tableBASE libraries

For BDAM/QSAM libraries, the default block size of 3120 results in 15 blocks per track and 82.6% utilization of the space. Increasing block size to 3174 results in 84% utilization (still at 15 blocks per track). Increasing it to 27998 (2 blocks per track) results in 98.8% utilization. Reducing it to 820 (39 blocks per track) results in 56.4% utilization.

tableBASE library space usage

tableBASE attempts to use library space efficiently. If the BLKSIZE (RECLEN) is larger, it fits more directory entries into each block. But every table definition requires 1 full block for the definition.

The actual data for each table is stored in an integral number of full blocks, with the minimum being one. A very small table (e.g., 10 rows with a row length of 12 bytes) still requires a full block, which results in a very low utilization, even if a blocksize of 800 (the minimum allowed) is used. On the other hand, a very large table (e.g., 50000 rows with a

row length of 200 bytes) will have over 99% utilization of its allocated space no matter what the blocksize.

FK for duplicate keys

FK for duplicate keys using binary search on a sequential table will always return the first row in the table which has the duplicate key.

Optimizing table access

Table access can be optimized by using a separate command area for each table. Instead of having to search a TSR directory each time to determine where the table resides in memory, tableBASE uses a reserved field in the command area (table handle) pointing to the position of the table in the TSR. By keeping a separate command area for each table accessed, this table handle is used to quickly locate a previously accessed table.

High-speed processing without EDF

When using EDF to trace an application in CICS, all calls to any tableBASE entry point will be traced using the tableBASE CICS Resource Manager. If you stop using EDF with the transaction, tableBASE will bypass the Resource Manager and automatically revert to high-speed processing.

Date-sensitive processing

With Version 6, the default date used by Date-Sensitive Processing rolls over at midnight. This will prove useful for 24x7 applications that rely on Date-Sensitive Processing.

Access Register mode

Version 6 has the following MVS restrictions applicable to AR mode (this applies only to applications coded in Assembler):

- tableBASE must be called in a primary mode, failure to do this may result in unpredictable results.
- tableBASE zeroes ALL access registers (A0-A15) on return to a caller. It is a responsibility of the caller to save and restore its access registers.

VSAM LSR pools for tableBASE libraries

tableBASE must be allowed to control its own I/O activity to ensure the integrity of library directories. If a VSAM library has been associated to an LSR pool, when the library is opened by tableBASE and found to be pooled, it is closed by tableBASE,

removed from the pool, and reopened. The exception to this is if the library has DISP=OLD, then it will be left in the LSR pool.

You risk destroying your data if you are using a tool that dynamically modifies the VSAM LSR setting. This applies to all releases of tableBASE.

When a tableBASE library is set to DISP=SHR, there is an exposure when using third party software like MAINVIEW® Batch Optimizer (MBO) from BMC Software, Inc., to dynamically turn on LSR processing (or mechanisms equivalent in effect). In these cases, tableBASE is not notified that LSR is turned on and cannot remove the library from the LSR pool. If more than one region can update the library, the library will be corrupted.

The library directory caching feature of Version 6 should provide equivalent or better results than LSR pooling.

Considerations for working with large tables

With Version 6, there are new considerations for dealing with large tables in tableBASE. Both the local TSR and VTS-TSRs are now in Data Spaces. The maximum size of a Data Space is 2G.

In addition you should consider the region size, the number of tasks running in the region, processing sequence, and the frequency and type of access activity. The choice of one search strategy over another can impact system performance.

What is large? For the purposes of the following recommendations, a table is considered very large when the total size is 50 MB or larger, or the number of records is greater than 1 million. The maximum row size a table can have is 32 KB (the maximum positive value in a half-word).

Recommended

- Binary or Hash search methods for tables with more than 300-400 rows.

Not recommended

- High frequency of Inserts with low expansion factor.

Note: With the introduction of Version 6, all tables are now treated as Pointer tables and have an Index, even if they are listed as being True tables. The change is transparent to the user, but Pointer and True tables are now treated internally as Pointer tables.

TSR

The TSR is in a Data Space in Version 6. Rollouts of tables (using TBTSLIB) is no longer supported with this release. The Data Space must be large enough to hold all tables that were paged tables in prior releases.

Binary searches

With Version 6, all searches cache the first 4 bytes of the key. This greatly improves search performance. With a large table where these 4 bytes may be the same throughout all rows of the tables these benefits may not be realized. Better results can be achieved by using a key that varies within the first 4 bytes.

Open/Store activity

The size of the table determines the load on the I/O system, possibly causing slowdowns in an online environment.

Creation method

For a keyed table, creation in sequential order can be much more efficient than in random order.

If created in random order, consider defining the table as Random initially, and after loading the Data Table, change the definition to Sequential.

Expansion factor

This controls the amount of additional memory that will be obtained to expand non-Hash tables. It should be set sufficiently large so all insertions between opens of the table will fit. For example, if the table has 100,000 rows, and a maximum of 10,000 rows are inserted with no deletions when it is opened, an expansion factor of 100 (10%) is sufficient.

tableBASE libraries

A tableBASE library will never use secondary allocated space because it can never increase in size dynamically. It will only use the initially allocated space. This initial space allocation does not need to be contiguous.

Linked tables and TB-PARM

If every tableBASE call in your applications are to VTS-TSR, then using the VTSNAME in the TB-SUBSYSTEM field in the TB-PARM to access VTS-TSR is more efficient than using the VTSFIRST, VTSLAST or using the ML command. Use of the VTSNAME in the TBPARM completely bypasses the use of the local TSR.

If you are using VTSFIRST, VTSLAST or ML to add a VTS-TSR to the LIB-LIST, the transaction process will first go to the local TSR (through CICS, IMS or batch) to find a handle (pivot entry) for the table requested, and if it is the first request for that table, a dummy entry will be entered in the local TSR. Then the process will go to the VTS-TSR to locate the table entry. The table entry address in the VTS-TSR would then be used to

update the initial dummy entry in the local TSR. On subsequent passes, the local TSR would have a valid address as a pointer to the VTS-TSR for that table. This process describes a linked table. For more information, see the *tableBASE Release Notes*. Also see "Linked Table" on page 27 and "Linked tables" on page 32.

Note: Use of VTSFIRST, VTSLAST or using the ML to command to access a table in VTS-TSR (linked table) does not allow for update commands against the table. Updates of the table can only be achieved by the use of the VTSNAME in the TB-PARM to access the table.

Performance and TB-PARM

DataKinetics recommends using a TB-PARM on every call to TBLBASE. If a TB-PARM is not used, performance can be significantly affected in some instances—for example, when DK1TCALL or DK1TROTB is in LPA or an authorized library.

Application performance with TBASEV or TBCALLV

To improve the performance of applications using the V5 TBASEV or TBCALLV interface, relink the application modules with the V6 TBASEV or TBCALLV (now aliases for DK1TCALL).

Limitations of duplicate key support

Duplicate keys can be created in several ways: Tables can be created with duplicate keys, alternate indexes can generate duplicate keys, and a change to an existing table definition can generate duplicate keys.

The duplicate key will be inserted in the table as a group. The group of duplicate keys will be sorted into the correct location in the table.

While all rows will be returned in order by key, if the table is processed by GN processing, there is no guarantee for the order of rows within the group of duplicate keys.

Appendix B

Compatibility with previous releases of tableBASE

tableBASE has been continually enhanced over the years. New features have been added, and existing features have been improved. Despite the many enhancements made, applications written for all releases of tableBASE since Release 4.2 will run under Version 6 – with no changes necessary. The applications may not be able to take advantage of some enhancements provided in newer releases, but they will run as is.

Many of these enhancements are significant, and justify your upgrading to this later release. By upgrading your applications to take advantage of these new features, you'll gain performance benefits as well added functionality. You'll find a summary of the enhancements made in Releases 5.0 and 5.1 at the end of this appendix.

The following sections describe some issues you should consider if you have not yet adopted the TBLBASE application programming interface (API) that was introduced in Release 5.0.

API changes

Version 6

The tableBASE API in Version 6, known as TBLBASE, can be used in CICS, batch, IMS TM, and DB2-SPAS operating environments. tableBASE programs that use this stub can be ported from one environment to another without requiring any stub relinking.

Release 5.0

From a programming viewpoint, the primary area of difference between releases is the API. Improvements have been made and parameters added or changed. The current TABLEBASE API was introduced with Release 5.0 and is designed to be used in all tableBASE environments – batch, CICS, IMS TM, and DB2-SPAS—using either a local TSR or a shared VTS-TSR.

Prior to Release 5.0, there were two "generations" of API. The older generation did not support a TB-PARM parameter and used a shorter COMMAND-AREA. This generation used the names TBCALL, TBCALLC, TBCALLV, and TBCALLI for the API; the names

varied based on the environment in which tableBASE ran. The other generation used a TB-PARM parameter, but it had fewer fields than the current TB-PARM. Again, there was a separate API – TBASE, TBASEC, and TBASEV – for each environment, and the COMMAND-AREA contained fewer fields than that used by TBLBASE.

While your older programs will run unchanged with Release 5.0 and above, you should not mix interfaces in the same program, for example, TBLBASE (Release 5.0 and above) and TBCALLC (pre-Release 5.0).

The following section contrasts the older COMMAND-AREA and TB-PARM parameters provided in earlier (pre 5.0) releases of tableBASE with the current versions.

COMMAND-AREA

In extending the COMMAND-AREA for Release 5.0, we simply added fields at the end and kept the first twenty-eight bytes unchanged. Here is the older COMMAND-AREA.

```
 01 xxxx-COMMAND-AREA-R4.
    05  xxxx-COMMAND               PIC XX            VALUE SPACES.
    05  xxxx-TABLE                 PIC X(8)          VALUE xxxx.
    05  xxxx-FOUND                 PIC X             VALUE SPACES.
    05  xxxx-INDIRECT-OPEN         PIC X             VALUE LOW-VALUES.
    05  RESERVED                   PIC X             VALUE LOW-VALUES.
    05  xxxx-ABEND-STATUS          PIC X             VALUE SPACES.
    05  xxxx-ERROR                 PIC S9(4          COMP VALUE +0.
    05  xxxx-COUNT                 PIC S9(8)         COMP VALUE +0.
*   The LOCK-LATCH must be specified for Online environments and VTS.
    05  xxxx-LOCK-LATCH            PIC X(8)          VALUE SPACES.
```

And, here is the later Version 5 / Version 6 COMMAND-AREA.

```
 01 xxxx-COMMAND-AREA.
   05  xxxx-COMMAND                      PIC XX   VALUE SPACES.
    05  xxxx-TABLE                       PIC X(8) VALUE xxxx.
    05  xxxx-FOUND                       PIC X    VALUE SPACES.
   05  xxxx-INDIRECT-OPEN                PIC X    VALUE LOW-VALUES.
    05  RESERVED                         PIC X    VALUE LOW-VALUES.
    05  xxxx-ABEND-OVERRIDE              PIC X    VALUE SPACES.
    05  xxxx-ERROR                       PIC S9(4)COMP VALUE +0.
    05  xxxx-COUNT                       PIC S9(9)COMP VALUE +0.
*   The LOCK must be specified for Online environments and VTS.
    05  xxxx-LOCK-LATCH                  PIC X(8) VALUE SPACES.
*   Release 5.x/6.0 command area extension
    05  xxxx-ROW-OVERRIDE-LENGTH         PIC S9(9)COMP VALUE +0.
    05  xxxx-ROW-ACTUAL-LENGTH           PIC S9(9)COMP VALUE +0.
    05  xxxx-FG-KEY-LENGTH               PIC S9(4)COMP VALUE +0.
    05  xxxx-FUNCTION-ID                 PIC S9(4)COMP VALUE +0.
   05  xxxx-FUNCTION-AREA                PIC X(28)VALUE LOW-VALUES.
   05  xxxx-DATE-AREA                    REDEFINES XXXX-FUNCTION-AREA.
       10  xxxx-DATE                     PIC X(8).
       10  RESERVED                      PIC X(20).
   05  xxxx-ABS-GEN-NO                   PIC S9(4) COMP.
   05  xxxx-ERROR-SUBCODE                PIC S9(4) COMP VALUE +0.
```


TB-PARM

As mentioned above, the TBCALL series of APIs do not use the TB-PARM parameter at all. The TBASE series of APIs uses a shorter TB-PARM parameter than does TBLBASE. This shorter TB-PARM is as follows:

```
01 TB-PARM-R4.
   10 TB-PARM-ID                          PIC X(2)  VALUE 'TB'.
   10 TB-FUNCTION                         PIC S9(4) VALUE +0.
   10 TB-RESERVED                         PIC X(4)  VALUE LOW-VALUES.
   10 TB-SUBSYSTEM                        PIC X(4)  VALUE LOW-VALUES.
   10 TB-RESERVED                         PIC X(8)  VALUE LOW-VALUES.
   10 TB-DATE                             PIC (8).
```

Not only is it shorter than the TB-PARM used with the current TBLBASE API, the above TB-PARM parameter also has a different format. Each field is described below.

1. TB-PARM-ID (2 bytes)

The literal 'TB' identifies this parameter as a TB-PARM communications area. (The Release 5.0 version of the TB-PARM parameter contains an identifying '5' in byte 5 for use by TBLBASE only; anything else in byte 5 indicates a previous release TB-PARM-R4 format for use by TBLBASE or any TBASEx interface).

2. TB-FUNCTION (halfword binary)

This field, when set to 0, indicates that the CALL does normal processing. When set to 1, it identifies the CALL as subject to Date-Sensitive Processing. In Release 5.0 and above, this field is now part of the COMMAND-AREA and is known as FUNCTION-ID.

3. TB-RESERVED (4 bytes)

This field is reserved for internal use and should be set to low values at initialization.

4. TB-SUBSYSTEM (4 bytes)

This field names the VTS subsystem for the target TSR. This should be set to low values or spaces to indicate a local (non-VTS) TSR. This field will be found in a different location in the Release 5.0 version of TB-PARM.

5. TB-RESERVED (8 bytes)

This field is reserved for internal use and is expected to be set to low values initially.

6. TB-DATE (8 bytes)

This field contains the application date to be used for Date-Sensitive Processing with TBASE. It is entered in YYYYMMDD format. For the TBLBASE API, this information appears in the Release 5.0 extended command area parameter, not in TB-PARM. It is known as xxxx-DATE (see previous section, COMMAND-AREA).

Release 5.0 enhancements

Here are the more important features of Release 5.0 that are of particular interest to programmers of tableBASE applications and tablesONLINE exit programs:

1. Migration towards new terminology. To reduce the learning curve for new users, some tableBASE terminology has been refined and modified to promote consistency with prevailing industry standards. Where appropriate, relational terms have been adopted in all descriptions of tabular structures. For example, the term "row" is used rather than "item". In tablesONLINE/CICS the terms View or View Table replace the earlier term, Field Definition Table (FDT). A new term, Table Object, refers to the named association of a View and a Data Table.

2. A common interface, TBLBASE, is available for batch, CICS, IMS, and VTS applications. This interface uses an optional new parameter, TB-PARM, as a general communications area and an extended xxxx-COMMAND-AREA parameter to provide greater functionality and flexibility. This interface offers significant performance improvements in CICS over the previous TBASEC interface. TBLBASE also provides date-sensitive processing capabilities.

3. Copybooks are now provided for all parameters of the API, to be used directly or, in some cases, as templates for developing table-specific parameters.

4. A new tableBASE command has been added: List Directory (LD) creates a directory listing in a table.

5. The capabilities of some existing commands have changed (OR, OW, DT, CD, FG, ML). The Open commands, for example, now allow for opening multiple Alternate Indexes against a Data Table (base or primary index table). The Open commands, OR and OW, are now fully compatible with Alternate Indexes. Definition commands DT, CD, and GD now provide for: century digits in dates, identification of a default View for the data, userid of last user to update the table, and user comments. Fetch Generic (FG) now supports table keys which contain asterisks, including binary or packed data. The ML command has been extended to support searching VTS subsystems as well as tableBASE libraries.

6. Table Access Tracking. A new field is maintained in a table definition to identify the last CICS/batch user to update the table.

7. Extensions to tablesONLINE/CICS User Exits: New exit points have been included in tablesONLINE to accommodate extensions to command line and row-level processing. Improved access to tablesONLINE system variables in EXITPARM allows for improved monitoring and control of tablesONLINE events. Backward

compatibility is supported to allow exits written for earlier releases of tablesONLINE to run in conjunction with new exits written for Release 5.0.

8. Creation or revision of a View (Field Definition Table) will cause the (optional) automatic generation of a COBOL FD copybook that includes a tableBASE Command Area.

9. Automated regeneration of Data Tables once their Views have been modified. This facility can be of assistance with the century date conversion of data files.

10. A new system table, TBOLM2M, is available to support automated many-to-many relationships. This table is used to assign Table Object names to related pairs of View name and Data Table name. In addition, the tracking of data and View combinations is also supported directly. Included with the definition of both data and View, is the option to specify the corresponding View or Data Table.

11. A new tableBASE library, TBDICLB, is available in CICS as a centralized repository for Views and dictionary-type tables.

12. TBDRIVC (called DK1TDRVC in V6) and TBDRIVER (called DK1TDRV in V6) command processors have been enhanced. A table name wildcard facility is provided for command processors to allow selected operations on a series of tables. Output from the new List Directory (LD) command in the CICS command processor, TBDRIVC, is scrollable, forward and backward. Maximum row size for TBDRIVx command processors has been extended to 32K. Batch TBDRIVER now accepts a parameter to specify the number of table rows to print. Several restrictions have been removed relating to date-sensitive processing.

13. Improved library performance. New internal structures improve the performance of directory searches and library management.

Release 5.1 enhancements

The enhancements incorporated into Release 5.1 of tableBASE focused primarily on improving the product's performance as well as maintaining compatibility with the latest versions of MVS.

Here are the more important features of Release 5.1 that are of particular interest to programmers of tableBASE applications and tablesONLINE exit programs:

1. VSAM libraries. MVS tableBASE libraries may now be defined as VSAM files. The BDAM file organization will continue to be supported. In fact, you may have some libraries defined as BDAM files and some as VSAM files. Or, you may define all of your libraries as either BDAM or VSAM files.

2. Faster initialization of BDAM libraries. Tests have shown that libraries are initialized under Release 5.1 in one-fifth the time of that used in Release 5.0

3. Reduced elapsed time to write tables to BDAM libraries in the batch, TSO, and IMS TM environments. In the CICS environment, elapsed time to write to BDAM libraries has been halved. These reductions are most noticeable when large tables are stored or rolled out to TBTSLIB.

4. More consistent JCL conventions across interfaces for specifying Release 5.1 Enhancements.

5. TBTSLIB overflow libraries. In Release 5.0, tableBASE does not verify if TBTSLIB is available and properly defined until it is required for overflow. In Release 5.1, this check is performed at first access to tableBASE to insure that a job or task does not abend halfway through processing. If you are migrating from Release 5.0, you were able to process without ever referencing TBTSLIB, if there was no need. In order to prevent untimely abends with the Release 5.1, the validation of TBTSLIB is always performed first. If tableBASE finds that TBTSLIB is not properly defined or initialized, the job or task will abend with a G040 or 0040, depending on the environment.

6. More use of storage above the line. tablesONLINE/CICS has been converted to COBOL II, thus tablesONLINE/CICS modules can be loaded above the line. The initial CICS GETMAIN for a transaction is now executed above the line.

7. Revised and improved PTF (Program Temporary Fix) management programs that make the application and tracking of tableBASE PTFs much more reliable and convenient.

Version 6 enhancements

Refer back to "What's new in Version 6" on page 21 for a list of Version 6 enhancements and modifications that are of particular interest to programmers.

For a thorough understanding of all enhancements included in this release of tableBASE, see the *tableBASE Release Notes*.

Appendix C

tableBASE run-time options

This appendix lists the tableBASE run-time option parameters. Most occur in all interfaces. If you are running applications across multiple environments any changes to these options in one interface may need to be made in each installed interface. Options can be customized for your site at installation time, using the TBOPTGEN macro in DK1Txx34 modules. Applications can further override options on an individual basis, using the TBOPT dataset.

CICSJRNL—CICS Journal File ID

This parameter pertains only to the CICS interface. It identifies the CICS Journal File onto which the TABLE SPACE Report records are to be written by the system, if strobe records are to be written. The delivered default value for the CICSJRNL parameter is 99.

This option applies to this region's users, regardless of which TSR they access.

DATERTNX—Date-Sensitive Processing Found Code

The DATERTNX parameter controls the value placed in the found code for a fetch by count (FC) operation on a date-sensitive table.

With an FC command, it is possible that the row retrieved is not within the date range. When this occurs, the found code is normally set to N to reflect the logical condition that the retrieved row is not a match. For compatibility with previous releases, the found code returned under this condition can be made to be X. The default setting of DATERTNX=N provides this compatibility with previous releases.

If Date-Sensitive Processing is being introduced in a new application, or no FC commands are used in the application, it is strongly recommended that DATERTNX=Y be specified so the found code will never be set to X. The delivered default is DATERTNX=N for backwards compatibility.

This option applies to this region's users, regardless of which TSR they access.

FGDELIM—Fetch Generic Delimiter

This parameter identifies the character to denote the end of the high order part of the key used in the FG (Fetch Generic) command.

The delivered default value for FGDELIM is an asterisk (*), for example, FG,table-name,key*. We recommend that programmers use the key length override function in the tableBASE command area as this executes faster, is less error prone and allows an asterisk to be part of the key.

This option applies to this region's users, regardless of which TSR they access.

Note: If you are installing tablesONLINE/CICS and wish to change the Fetch Generic Delimiter, please see the tablesONLINE/CICS section of the *tableBASE Installation Guide*.

HASH_HI_DEN_LIM—High Density Limit for Hash Indexes

HASH_LOW_DEN_LIM and HASH_HI_DEN_LIM limit the density of the index for a hash table opened in Version 6. These values are designed to prevent performance problems which can occur when inappropriately high values are used when defining hash tables (can result in numerous new indexes being created). Other problems occur if the difference between low and high density values is too small. A ratio of 2/3 is now enforced: Low density may not be greater than 2/3 of high density.

HASH_HI_DEN_LIM=nnn must be between 100 and 900 (10% and 90%); the default is 900. We recommend the value be lowered to 500.

This option applies to this region's users, regardless of which TSR they access. (An exception to this is a situation where a Version 6, Release 6.0.3, or higher, application accesses a Release 6.0.2 or lower VTS-TSR—in this case the region’s density limits are ignored.)

HASH_LOW_DEN_LIM—Low Density Limit for Hash Indexes

See above (“HASH_HI_DEN_LIM—High Density Limit for Hash Indexes”).

HASH_LOW_DEN_LIM=nnn must be between 10 and 600; the default is 600. We recommend the value be lowered to between 200 and 300.

This option applies to this region's users, regardless of which TSR they access. (An exception to this is a situation where a Version 6, Release 6.0.3 , or higher, application accesses a Release 6.0.2 or lower VTS-TSR—in this case the region’s density limits are ignored.)

LDS—LDS use

Specifies whether the VTS-TSR defined in this jobstep will be mapped to a LDS (Linear Data Set). The LDS must be pre-defined.

In a batch environment only, this option can be set to Y to allow a mapping of the VTS-TSR to the LDS. If set to N, there is no mapping to the LDS.

The delivered default is *N*.

This option applies to the TSR created by this region/job and all users accessing it.

Note: This parameter applies only if you have licensed the optional tableBASE Process Manager.

LIBnn, ML—tableBASE Library List

Specify in sequence, the names of the libraries (DDNAMEs) in the tableBASE Library List (LIB-LIST). In TBOPTGEN, this is specified as a comma delimited list under ML. In the TBOPT file, the library names are specified as LIBnn=lib_name. Up to 10 LIBnn may be specified, as long as the numbers are contiguous. For example, LIB01=MAINLIB, LIB02=TESTLIB.

A VTS-TSR may also be included in the search list by specifying the name of the VTS-TSR prefixed by "VTS:" in LIB-LIST (for example, VTS:DKL1), instead of a library DDNAME.

The delivered default is ML=MAINLIB which is equivalent to supplying a single parameter LIB01=MAINLIB. If no ML is set, tableBASE uses MAINLIB.

This option applies to this region's users, regardless of which TSR they access.

LISTOPTIONS—List Parameter Options

Determines whether or not to list execution time parameters. If LISTOPTIONS=Y is specified, all default parameters and parameters overridden by the TBOPT dataset are listed in the JESMSGLG. LISTOPTIONS=N suppresses this list. The delivered default is LISTOPTIONS=N.

LISTOPTIONS=X is a special setting for TBOPTGEN (DK1T1134). It is the equivalent of LISTOPTIONS=N if the TBOPT DD was not present in the jobstream, and the equivalent of LISTOPTIONS=Y if the TBOPT DD is present. LISTOPTIONS=X applies to TBOPTGEN (DK1T1134) only; it does not apply to TBOPT.

This option applies to this region's users, regardless of which TSR they access.

LOCKTIMERC—Lock Timer Wait Value

LOCKTIMERC=nnnnnn specifies the number of seconds (default 0) that tableBASE should wait for a lock. When the LOCKTIMERC interval has passed, RC=71 is returned. A value of LOCKTIMERC=0 specifies that the process will never time out.

The lock controlled by LOCKTIMERC is used internally by tableBASE to maintain table integrity in the TSR. It is unrelated to the table ENQUEUE that occurs when a table is opened for write (OW).

This option applies to this region's users, regardless of which TSR they access.

LOCKTIMEWTO—Lock Timer Message Wait Value

LOCKTIMEWTO=nnnnnn specifies the number of seconds (default 30) to wait before issuing a message (DK100227W) that the process is waiting for a lock. A value of LOCKTIMEWTO=0 specifies that no warning message will be issued.

The lock controlled by LOCKTIMEWTO is used internally by tableBASE to maintain table integrity in the TSR. It is unrelated to the table ENQUEUE that occurs when a table is opened for write (OW).

This option applies to this region's users, regardless of which TSR they access.

MAXNMTAB—Maximum Number of Tables

Determines the number of tables that can be opened in a given local TSR or VTS-TSR. (With the possibility of creating a TSR of up to 2G in size, there can be a great variation in how many tables will actually be in a TSR at any time.) If the value of the MAXNMTAB is not set by the user (or is set to zero) then a default value is calculated.

If an attempt is made to open more tables than is indicated by MAXNMTAB, an error code of 20 will be given: The maximum number of tables has been exceeded. Check MAXNMTAB.

This option applies to the TSR created by this region/job and all users accessing it.

Note: For a VTS-TSR region, VTSNMTAB is retained for backwards compatibility and is treated as an equivalent.

User sets the value of MAXNMTAB

To optimize the memory required for system overhead, the user may choose to set the value of MAXNMTAB rather than use the default value. MAXNMTAB can be set to any integer, zero or greater. However, if MAXNMTAB is set to a value greater than the

default for the given TSR size, the default is used and the user in notified with a warning message.

After the user sets the value of MAXNMTAB, memory is allocated and an index is created that contains enough entries to support the maximum number of tables indicated. Additional memory is allocated as tables are opened. This memory is reused as tables are closed and new tables opened. This approach keeps the system overhead in line with the number of tables that are opened.

The calculation of the default value of MAXNMTAB

If the user does not set the value MAXNMTAB or the value is zero, the tableBASE software calculates the default value based on the size of the TSR. Each table in the TSR occupies a minimum of 4K pages. The maximum possible number of tables is 51,455, for a 2G TSR; the default, when unspecified is one table for every K page in the TSR.

MTRETAIN—Retain Rows and Index Areas

MTRETAIN=Y|N determines whether the allocated rows and index areas are to be retained when an MT command is issued or whether they are to be reduced to the original allocation before table expansion. (It may be necessary to save the original allocations). The default is "N", which is current processing—not retained.

Also see "ZEROROWS—Zero rows" on page 403.

This option applies to the TSR created by this region/job and all users accessing it (applies only to Release 6.0.3 and above).

MULTITASKING—Multitasking

Specifies whether this region will allow multiple subtasks to access tableBASE. DB2 stored procedures are considered to be multitasking.

This option applies only to batch environments. All other environments are multitasking in their nature, and will not accept this option. The MULTITASKING option can be set to Y to allow multiple subtasks within a region to access tableBASE concurrently. Each subtask may have multiple concurrent subtasks of its own, with no limit to the level of multitasking. The delivered default is N (multitasking not enabled).

This option applies to this region's users, regardless of which TSR they access.

Note: If a batch application attempts to issue a tableBASE command from a second TCB with Multitasking=N, it will abend with an S0C3 and display message: "DKL00496E Multitasking requires TBOPT Multitasking=Y be set".

Note: Please call tableBASE Technical Support for assistance when writing applications that access tableBASE in a multitasking environment.

MULTOPNX—Multiple Alternate Index

MULTOPNX is a flag indicating whether to allow Multiple Alternate Indexes to a Data Table to be open concurrently. Releases of tableBASE prior to Release 5.0 allowed only one Index to be open. Subsequent Indexes were invoked using the IA command. For more information about Alternate Indexes, see the *tableBASE Concepts and Facilities Guide* (Indexes) and the *tableBASE Programming Guide* (various).

This option is maintained for compatibility with previous releases, and is not used in tableBASE Version 6. The only valid value for this flag is Y and this is the delivered default.

This option applies to this region's users, regardless of which TSR they access.

Note: For assistance with this default value, please contact tableBASE customer support.

OVRRIDES—Allow Changes to Status Switches

The Status Switches may be changed by the application with the use of the Change Status (CS) line command. Overriding individual Status Switches may be inhibited by the settings of the override controls. Overrides are specified as a string of y's and n's; for example: `yyynynnn`. All eight positions must be specified.

The defaults for these values will depend upon the interface, as listed in Table C-1.

Table C-1: Overrides default values

Position	Default	Override setting	Meaning
1	Allow Change To Abend On Errors	Y	The application is allowed to set the ABEND status switch.
		N	The application is not allowed to set the ABEND status switch.
2	Allow Change To Wait For Enqueued Tables	Y	The application is allowed to set the WAIT status switch.
		N	The application is not allowed to set the WAIT status switch.
3	Allow Change To Return Empty Rows In Hash Tables	Y	The application is allowed to set the HASH-EMPTIES-RETURNED status switch.
		N	The application is not allowed to set the HASH-EMPTIES-RETURNED status switch.

Table C-1: Overrides default values

4	Allow Change To Permit Implicit Open Of Tables	Y	The application is allowed to set the IMPLICIT OPEN status switch.
		N	The application is not allowed to set the IMPLICIT OPEN status switch.
5	Allow Change To Trace tableBASE Commands	Y	The application is allowed to set the tableBASE TRACE status switch.
		N	The application is not allowed to set the tableBASE TRACE status switch.
6	Reserved for future use; specify as N.		
7	Reserved for future use; specify as N.		
8	Reserved for future use; specify as N.		

These options apply to this region's users, regardless of which TSR they access.

SWITCHES—Status Switches

The Status Switches are a series of one-byte codes for altering the operation of the Application Programming Interface used by your programs when requesting a tableBASE service. Switches are specified as a string of y's and n's; for example: `yyynynnn`. All eight positions must be specified. For information about switch settings, see Table C-2.

The switches are stored in the parameter module for the interface. As each batch job or online task begins, the run-time values will be set according to the values in the parameter module but may be changed using the values in the TBOPT file. For more information, see Parameters in the MVS Batch section of the *tableBASE Installation Guide*.

The application also may change the values with the CS, the Change Status command (unless changing a particular switch is suppressed via the override controls).

The defaults for these values depend upon the interface and are described in the *tableBASE Installation Guide* in the appropriate chapter:

- tableBASE MVS batch
- CICS Interface option
- IMS TM Interface option
- Virtual Table Share option.

Table C-2: Switches default values

Position	Default	Switch setting	Meaning
1	Abend on Errors	Y	Abend processing is to be performed on tableBASE errors 0001-0099 or 1001-1099. Errors 100 to 999 and errors greater than 1099 always abend.
		N	Abend processing is not to be performed on tableBASE errors 0001-0099 or 1001-1099. User programs will handle these return codes.
2	Wait for Enqueued Table	Y	tableBASE is to wait for tables that are enqueued
		N	tableBASE is not to wait for such enqueued tables. In this case tableBASE will terminate with a user 0072 abend if abend processing is enabled, or, will return an error code of 0072: TABLE UNAVAILABLE; NO WAIT in the command area if abend processing has been disabled.
		Note: Waiting in an online environment is normally discouraged. In CICS and IMS, the default is set to N	
3	Return Empty Rows in Hash Tables	Y	tableBASE is to return empty rows for hash tables
		N	tableBASE is to suppress the return of empty rows for hash tables
4	Allow Implicit Open Of Tables	Y	tableBASE is to automatically open tables for read on first access
		N	tableBASE is to suppress automatic opens; an explicit open command, OR, OW, or IA must be issued to open a table
		Note: We recommend setting Allow Implicit Open of Tables to N, especially in multi-user environments. This gives better control over which tables are opened.	

Table C-2: Switches default values (Continued)

Position	Default	Switch setting	Meaning
5	Trace tableBASE Commands	Y	tableBASE automatically records the last ten commands executed per thread. This is done for diagnostic purposes. Contact DKL for more information.
		N	tableBASE does not trace commands
6	Reserved for future use; specify as N.		
7	Reserved for future use; specify as N.		
8	Reserved for future use; specify as N.		

These apply to this region's users, regardless of which TSR they access.

STROBE—Strobe Interval

Specify a numeric value, from 0 to 2,147,483,647(2G – 1) to control the intervals at which Table Space statistics report records are to be generated. After the specified number of calls to tableBASE that access the TSR the strobe reporting routine is called.

The delivered default is 0.

A final strobe report will always be produced if the TBTSRPT DD statement is present (even if set to 0). In order to suppress strobe reporting, the TBTSRPT DD statement must be removed from the job. For strobe=0, no intermediate strobe report is produced.

This option applies to the TSR created by this region/job and all users accessing it.

Note: The numeric value must be specified without commas.

TABLEWAITRC—Table open enqueue wait time

Specify a value to indicate the number of seconds that a user will wait to obtain the MVS enqueue to open a table for read or write before timeout. If the enqueue is not obtained before the timeout, tableBASE will return code 72, or will abend, if the *Abend on Errors* switch is set to Y.

The delivered default is 0 (wait forever).

The Wait for Enqueued Table switch (see Table C-2 on page 396) must be set to Y for this parameter to have any effect.

This option applies to this region's users, regardless of which TSR they access.

TABLEWAITWTO—Table open enqueue report time

Specify a value to indicate the elapsed time before a message will be generated to report that the enqueue has not yet been received.

The delivered default is 30 (no messages).

The Wait for Enqueued Table switch (see Table C-2 on page 396) must be set to Y for this parameter to have any effect.

This option applies to this region's users, regardless of which TSR they access.

TPVM—TPVM use

Specify a valid VTS Manager (TPVM) name to use for all accesses to VTS-TSRs in this jobstep. The name must be pre-defined, and can be up to eight characters in length.

The delivered default is *compat*.

This option applies to the TSR created by this region/job and all users accessing it.

Note: This parameter applies only if you have licensed the optional tableBASE Process Manager.

TSR_ALGORITHM

TSRAlgorithm=P|D allows the TSR Space Manager to determine if TSR space usage should be maximized or performance should be optimized when TSR space is allocated and deallocated for table usage. The TSR space manager is the tableBASE component responsible for tracking TSR space usage and allocating and deallocating TSR space when tables are opened, closed, expanded or reduced in size.

The following is a description of each option:

- P (Performance) indicates that the TSR will be optimized for performance. Space will be assigned to tables within the TSR so as to minimize CPU usage, which may result in a less than optimum use of space.
- D (Default) indicates that there will be no optimization, and there will be no messages regarding optimization. However, messages will be provided regarding current percentage of TSR capacity.

The delivered default is D.

This option applies to the TSR created by this region/job and all users accessing it.

TSR_WARNING_FREQ

Specify a value to indicate the frequency of reports generated when the TSR allocation percentage (as defined by TSR_WARNING_PCT) is exceeded. 0 results in a message for every allocation over the specified percentage; 30 results in a wait of at least 30 seconds before repeating the message; 999 results in a wait of just over 15 minutes before repeating the message.

The delivered default is every one second (001).

This option applies to this region's users, regardless of which TSR they access. (Different regions accessing a VTS-TSR can set their own thresholds.)

TSR_WARNING_PCT

Specify a value to indicate the percentage of TSR allocation allowed before a message will be generated to report that the allocation percentage has been exceeded. The message will be repeated on every allocation of this percentage, subject to the TSR_WARNING_FREQ parameter.

The delivered default is 85 (85%).

This option applies to this region's users, regardless of which TSR they access. (Different regions accessing a VTS-TSR can set their own thresholds.)

TSRACCESS—R-O or R/W

Specifies whether the VTS-TSR defined in this jobstep will have Read-Only access or Read/Write access. This option can be set to RO to for Read-Only access; or RW for Read/Write access.

The delivered default is *RW*.

This option applies to the TSR created by this region/job and all users accessing it.

Note: This parameter applies only if you have licensed the optional tableBASE Process Manager.

TSRSIZE—tableSPACE Region Size

The TSRSIZE parameter is an integer representing the amount of storage to be used for the TABLE SPACE REGION (TSR). Version 6 uses Data Spaces for all TSRs, whether local or VTS. This ensures that tableSpace memory requirements do not affect other memory requirements in the region. The format for TSRSIZE is:

- nnnnnnnn = bytes
- nnnnnnnK = kilobytes

- nnnnM = megabytes
- nG = gigabytes

Value recommendations

The delivered default value for the TSRSIZE is 10M for all interfaces. The current minimum size of a TSR is 28 KB (but this may change in subsequent maintenance releases). The maximum size of a TSR is 2G (or 2048M or 2097152K). TSRSIZE only accepts up to eight digits, thus you cannot specify 2G as TSRSIZE=2147483648.

Note: If you specify a value of 0, the minimum size will be substituted; if you specify a value of between 1 and the minimum, an error code will be returned.

Other considerations

If you are upgrading from an earlier release, you must ensure you are allocating enough space for all tables open simultaneously in the region. A conservative TSR size for Version 6 would be the sum of the prior release TSR size and the size of TBTSLIB (the Rollout library from Release 5).

If you are upgrading from an earlier tableBASE release, you can determine the current TSR size in one of two ways:

1. use the LT command in a tableBASE driver program—batch program DK1TDRV / TBDRIVER, CICS transaction TBDR,

2. browse strobe reports produced by your currently-installed tableBASE release.

Failure to set your TSR size parameter appropriately can produce the following error:

Error 92 There is insufficient tableSPACE region available.
Increase TSR size.

Notes:

1. TSREGION and TSR parameters are maintained for backwards compatibility and are treated as equivalents to TSRSIZE.

2. There is a possibility that your system programmers have established limits on the maximum size or the maximum number of Data Spaces. All local TSRs are allocated as SCOPE=SINGLE Data Spaces; VTS-TSRs are allocated as SCOPE=ALL Data Spaces.

3. For users upgrading to Version 6, batch now requires the use of the TSRSIZE parameter. It is no longer sufficient to use the REGION= parameter in the JOB, or any other method. You can use the same parameter value for TSRSIZE that you did

for REGION. For instance, if the REGION value was 32M or less, the batch job could have getmained up to 40M of space to hold tableBASE tables (32M above the line, and 8M below). If the REGION value was larger than 32M, the specified value plus 8M should be sufficient. Otherwise, see the recommendations above.

This option applies to the TSR created by this region/job and all users accessing it.

USEREXITS—User Exits

Specifies whether User Exits are used. Setting the tableBASE execution-time parameter USEREXITS=Y causes the DK1TX072 module to be dynamically loaded during the initialization of tableBASE. For more information, see *tableBASE User Exits* in the *tableBASE Administration Guide*.

This option applies to this region's users, regardless of which TSR they access.

VTSFIRST, VTSLAST—VTS Search Sequence

VTSFIRST and VTSLAST are two parameters that can be used to control the sequence in which tableBASE searches libraries when tableBASE VTS is installed.

VTSFIRST is the name of the VTS-TSR that is to be searched for tables before searching any libraries listed in the tableBASE library-list. VTSLAST is the name of the VTS-TSR that is to be searched for tables after searching any libraries listed in the tableBASE library-list.

If you wish to override the installation default for this parameter, it is strongly recommended that you use the TBOPT DD statement described later in this chapter to do so. If you set the VTSFIRST parameter by tailoring the installation parameters at installation time, problems may occur. For example, any processes that operate on the library copy of a table that is open in VTS may abend with a tableBASE error code of 3 or may return unpredictable results.

Note: If VTSFIRST or VTSLAST is specified in the default parameters (DK1Txx34), setting VTSFIRST=0 or VTSLAST=0 in the TBOPT dataset for the job turns the respective parameters off.

The delivered defaults are VTSFIRST, VTSLAST set to null strings.

This option applies to this region's users, regardless of which TSR they access.

Note: These parameters apply only if you have licensed the VTS Agent (the optional tableBASE VTS component). Do not change the delivered defaults supplied for these parameters if you have not licensed the VTS Agent.

VTSNAME—specifying the name of a VTS-TSR

The VTSNAME option specifies the name to be assigned to the VTS-TSR when it is initialized by the VTS Agent. The name is used by other regions to access the shared TSR.

Notes:

1. Starting in Release 6.0.3, VTSNAME can be 1 to 8 characters. The naming restrictions are identical to those for DDNAMEs in IBM JCL. VTSNAMEs of 1 to 4 characters created in prior releases are still supported.
2. Care must be taken when assigning VTSNAMEs, particularly if VTSNAMEs are to be used in the ML List— the combined length of the VTSPREFIX and the VTSNAME cannot exceed eight characters (see "VTSPREFIX" below).
3. For compatibility with previous releases, where it was specified in the TBOPTV dataset of regions accessing tableBASE, VTSNAME can be used to specify the default VTS-TSR accessed by the TBCALLV and TBASEV interfaces. Note that the tableBASE options for applications using TBCALLV and TBASEV may have changed. See the "Environmental interfaces" chapter in the *tableBASE Programming Guide* for additional information.

This option applies to the TSR created by this region/job and all users accessing it.

VTSPREFIX

The VTSPREFIX option specifies a prefix that will be used by tableBASE to denote a VTS name within the ML list.

The delivered default is *VTS:*. There can be 0-3 characters before the colon; the colon is mandatory. Example valid values:

- VTS:
- V:
- :

The maximum number of characters in total for the VTSPREFIX and the actual VTS name is eight. Therefore, a VTS name of eight characters could never be used in conjunction with the ML command. Example valid combinations:

- VTS:SHAR
- V:SHARE1
- :SHARE99

This option applies to this region's users, regardless of which TSR they access.

ZEROROWS—Zero rows

Related to the MTRETAIN parameter (see "MTRETAIN—Retain Rows and Index Areas" on page 393)—applies only when MTRETAIN = N.

ZEROROWS=Y|N determines whether the data rows area should be zeroed when it is deallocated. The default is Y, which is current processing. Note that the index area is never zeroed, except for hash indexes.

This option applies to the TSR created by this region/job and all users accessing it (applies only to Release 6.0.3 and above).

Appendix D

TBOPT dataset coding

The TBOPT dataset can be a sequential file, a member of a data set, or, for CICS, a VSAM dataset. The TBOPT dataset can be specified for all interfaces, including VTS. The dataset must contain fixed-length 80-byte records.

Note: The TBOPTV functionality has been integrated into TBOPT, allowing for a single source of run-time parameter input. TBOPTV is still maintained for backwards compatibility. If both TBOPT and TBOPTV are used, TBOPT is read first.

The data in TBOPT uses the same parameter names and values as are coded on the TBOPTGEN macro for the defaults, with the exception of LIB-LIST. TBOPT uses LIBNN to specify tableBASE libraries to update the tableBASE Library List.

Each parameter is entered on a single line in the dataset. The parameter may begin in any column. A line beginning with an asterisk (*) denotes a comment. Comments may also be added after the parameter value. A semicolon may be used to indicate line end. Comments may follow the semi-colon.

Although each region may have defined its own TBOPT dataset, all regions can share a sequential DASD dataset, and CICS regions can share a VSAM TBOPT dataset.

A sample TBOPT dataset for a batch region follows:

```
//TBOPT    DD *
* A leading asterisk denotes a comment
  ListOptions=Y
  TSRegion = 12M
  MAXNMTAB=500
  LIB01 = TESTLIB
  LIB02 = MAINLIB
/*
```

Note: ListOptions=Y is handy for diagnostic purposes; TESTLIB is first for batch testing.

Note: With the exception of the LISTOPTIONS parameter, parameters must appear only once in the TBOPT file. The form KEYWORD=* indicates that the site default is to be used. The form KEYWORD=0 indicates that the default value of a parameter that takes a character string be nullified.

Appendix E

tableBASE Messages

This appendix contains all of the DataKinetics tableBASE error codes and messages that can be encountered during the normal installation, administration and operation of the product. The error codes and messages fall into the following categories:

tableBASE error codes—error conditions that may be encountered while running tableBASE (see "tableBASE error codes" on page 408).

tableBASE messages—conditions that may be encountered with running tableBASE in conjunction with the tableBASE VTS agent, CICS, batch, IMS, and other programs/ utilities (see "tableBASE messages" on page 418).

TBEXEC error messages—conditions that may be encountered while running tableBASE in conjunction with the TBEXEC batch utility program (see "TBEXEC error messages" on page 425).

tableBASE Process Manager messages—conditions that may be encounted while running tableBASE Process Manager (see "tableBASE Process Manager messages" on page 430).

tablesONLINE/CICS error messages—conditions that may be encountered while running tableBASE in conjunction with the tablesONLINE/CICS application (see "tablesONLINE/CICS error messages" on page 444).

tablesONLINE/ISPF error messages—conditions that may be encountered while running tableBASE in conjunction with the tablesONLINE/ISPF application (see "tablesONLINE/ISPF error messages" on page 460).

Other messages—conditions that may be encountered while tableBASE performs user initiated activities such as library conversions (see "Other error messages" on page 468).

tableBASE error codes

The following tableBASE return codes are returned by tableBASE as a result of events detected in the processing of commands. In CICS these codes are prefixed by U to indicate a user application error. Additional errors generated by VTS-TSR (prefixed by TBV1) and tablesONLINE (prefixed by TB-) may also be encountered. If the *Abend on Errors* switch is set to 'N' the return code and subcode are returned to caller in the command area. If the the *Abend on Errors* switch is set to 'Y' the thread will abend, and a message will be written to the JES log with prefixes, etc. Below is an example:

```
THE SPECIFIED LIBRARY IS NOT SUITABLE       RC = 0060 SC = 0006
```

In this example, you would reference 60-06 in the following table. Depending upon the error condition, there may or may not be a subcode.

Note: In the event the application thread abends, the tableBASE error code is used as the user abend value, and any associated error subcode is used as the abend reason code.

Table E-1: tableBASE messages and error codes

Return Code	Text	Meaning / Instructions
00-00	tableBASE command executed successfully	No action is indicated.
00-01	Hash Low and/or High Density values adjusted based on limits	tableBASE command executed successfully. Table hash density limit(s) were substituted. High density was greater than HASH_HI_DEN-LIM or Low Denisty was less than HASH_LOW_DEN_LIMI. Low density value may not be greater than 2/3 of the high density value.
00-03	The table specified on the open command was already open.	tableBASE command executed successfully. Table specified was already open in the designated TSR.
01-00	The specified command is invalid.	The two-character Command code specified is not a valid tableBASE command.
02-00	The specified table must be open for this command.	Open the table before processing this command.
02-01	The table is in the process of being opened by another process.	This table is being opened and this command cannot wait for the open to complete. Retry the request.
02-98	Invalid parameter(s) were specified for the CL command.	Correct the parameters and resubmit the command (the CL command does not take any parameters).
03-00	The table is not closed.	The table must not be open in the TSR for this command.
03-01	The command requires that the table be closed	Close the table in the TSR and resubmit the command.
03-02	The table specified by the LD command has an incorrect row size.	Correct the row size for the LD table and resubmit the command. The row size depends on the DIRTYPE specified; see the *Programming Guide*.
03-03	The table specified by the LD command has an incorrect key size.	Correct the key size for the LD table and resubmit the command. The key size depends on the DIRTYPE specified; see the *Programming Guide*.
03-04	The table specified by the LD command has an incorrect key location.	The key location for the table created by the LD command is always 1. Correct the table definition and resubmit the command.
03-05	The table specified by the LD command is an alternate index.	The table used by an LD command must be defined as a data-table. Correct the table type and resubmit the command.

Return Code	Text	Meaning / Instructions
03-06	The table specified by the LD command is a linked VTS table	The table used by an LD command cannot be a linked VTS table.
03-07	The library specified by the LD command cannot be a VTSNAME	The LD command only works with tableBASE libraries on DASD. LD command usage with a VTS-TSR in the library list is not supported. Use the LT command to list tables in a VTS-TSR.
03-08	The table is already open from a library with a different DSN	The table cannot be opened from the specified DSN. It is already opened in the TSR from a different DSN.
03-09	The table is in the process of being opened by another process.	This table is being opened. Retry the request after closing the table.
04-00	The FG command is not allowed; the table is not sequential (S or D)	Change the table organization to sequential (S or D) and resubmit the command.
05-00	The indirect open search criterion was not found in the primary table	The indirect open search criterion set by the SI command did not match any keys in the primary table specified on this command. See the *Programming Guide* for details on using INDIRECT-OPEN.
06-00	The count specified is invalid	Correct the COUNT value in the COMMAND AREA and resubmit the command.
06-01	The COUNT must be a positive number.	The COUNT value in the COMMAND AREA cannot be negative. Correct and resubmit the command.
06-02	The COUNT exceeds the number of rows	The COUNT value in the COMMAND AREA cannot exceed the number of rows in the table (except for hash tables). Correct and resubmit the command.
06-03	The COUNT points to an empty row in a hash table.	The COUNT value in the COMMAND AREA for RC and DC commands for a hash table must point to a valid row.
07-00	The library DSN or DDNAME has been changed. Use DV or DW before STore.	If the library DSN or DDNAME for an open table has changed, the DV or DW command must be issued for the table before the table can be STored.
08-00	The generation number specified for this table is invalid.	Correct the generation number specified and resubmit the command. Generation numbers can be specified as absolute numbers from 1 to 255 or as negative numbers relative to current generation. The current generation is 0. (See the *Programming Guide* for details on specific commands)
09-00	The table is not found.	The table is not open in the TSR and cannot be found in the current library list (ML-list).
09-01	The table could not be found in any of the libraries in the ML list.	Ensure the ML-list includes the library where the table can be found and resubmit the command.
09-02	The table specified could not be found in the linked VTS-TSR.	The table search is linked to a table in a VTS-TSR and the table is not open there.
09-03	The table specified could not be found in the designated library.	Either ensure the ML-list includes the library where the table resides or ensure the table is resident in the currently designated library. If the command specifies a specific library DDNAME, ensure the table exists on that library.
10-00	The STATUS-SWITCHES parameter provided for the CS command is invalid	Correct the STATUS-SWITCHES parameter and resubmit the command. (See the *Programming Guide* for details on specific switches)
11-00	The Number of Generations to Keep in the DT parameter must be 1 to 9.	tableBASE supports up to 9 generations of a table on a library. The number of generations to keep in the DT parameter must be 1 to 9. Correct the value and resubmit the command.
12-00	The table name specified is invalid.	A valid table name is a string of 8 bytes that are not all blanks, all low values, all high values, or :TMPNAME. Please refer to the *Programming Guide* for more details. In some applications, such as TBOL/CICS, more restrictive rules may be implemented.

Return Code	Text	Meaning / Instructions
13-00	The command is invalid in this environment	The two-letter command code is valid. It is not supported in this execution environment.
13-01	This command is only supported for a VTS Agent.	The strobe-request internal command is only supported in the VTS Agent.
13-02	The DL (Define Library) command is not supported in CICS.	The DL command is only supported in the Batch interface.
13-03	Library expansion process may not be initiated from CICS.	The library expansion process is only supported by the TBEXEC (DK1TEXEC) Batch utility running in a non-multitasking environment.
13-05	Updates not permitted on a linked VTS table.	Only READ access is permitted on a table linked to a table in a VTS-TSR. (See the *Programming Guide* for more details on linked tables)
13-06	Update commands are not permitted when a TSR is read only.	Update commands are not permitted when TSRACCESS=RO (TSR is read only). This option is supported only with the TPM product.
14-00	The ROW-SIZE specified is invalid. Must be from 1 to 32767.	The row size specified must be from 1 to 32767. Correct it and resubmit the command.
15-00	The KEY-SIZE specified is invalid. Must be from 1 to 256 and fit in row.	The key size specified must be from 1 to 256, and must fit in the row based on key location. Correct the key-size and resubmit the command. Note that date-sensitive processing can affect this.
16-00	The KEY-LOCATION specified is invalid. Must be from 1 to row-size.	Correct the key-location specified and resubmit the command. Note that the key location must allow the key size to fit within the row size.
17-00	The key will not fit within the row.	The key will not fit within the row because either the key's location in the row or its length is beyond the bounds of the row. Correct the key-size/key-location and resubmit the command. Note that date-sensitive processing can affect this.
18-00	Insufficient space is available on the tableBASE library.	The library must be enlarged. A table cannot be stored on the library because there is insufficient free space available. Note that during the processing of a Store command there must be room in the library to hold 2 copies of the table on the library. There are 2 options: 1) create a new larger library and copy the existing library into it, or 2) expand the library. Both options are supported by the Batch Utility DK1TEXEC (TBEXEC). (See the *Batch Utilities Guide* for details.)
19-00	This table already exists on the target library.	A new table cannot be STored on a library when it already exists on that library. Preceding the STore with a DW command will allow the table to overlay the previous table.
20-00	The maximum number of open tables has been exceeded. Check MAXNMTAB.	The MAXNMTAB option for the region must be increased to allow more tables into the TSR. Temporarily, tables can be CLosed. To change this value permanently, override the value in a TBOPT DD or update DK1Txx34. (See the *Installation Guide* for the latter option.)
21-00	The Organization specified is invalid, must be blank, R, U, S, D, H.	Correct the organization specified and resubmit the command.
28-00	Paged tables are not supported in V6.	Paged tables are not supported in Version 6. All paged tables were converted to resident during the upgrade to V6. Storage Mode Code (SMC) of "P" (for Paged) is no longer allowed on the DT or CD command.

Return Code	Text	Meaning / Instructions
29-00	The secondary table is not located on same library as primary table	When the Open Indirect option is invoked for an Open command, the tables to be opened must reside in the same library as the primary table. (See the *Programming Guide* for details on Open Indirect, or the SI command for more details.)
30-00	The password supplied is invalid.	Accessing this table on a tableBASE library requires a valid password. Supply the correct password and resubmit the command.
31-00	The write password is either missing or incorrect.	Opening this table for write access from this tableBASE library or updating it on this tableBASE library requires the Write Password. Correct the password and resubmit the command.
32-00	The ST command requires that the table be opened for write.	A table can be STored on a tableBASE library only if it is Open for Write in the TSR. Execute the OW command before attempting to STore the table.
33-00	A different generation of the table is already open.	Two generations of the same table cannot be open in a TSR at the same time. Close the table and open it again in the region in order to access the latest generation on the library.
39-00	The RN command failed. The new name already exists in the library.	The table rename (RN) command failed because a table with the new name already exists in the library.
40-00	The library DDNAME does not exist.	A library name specified in the library list (ML-list) or a command parameter does not correspond with an allocated DDNAME.
40-01	The specified library DDNAME is not allocated.	Correct the library names in the library list (ML-list) or the command parameter and/or allocate the DDNAME in the region (add a DD statement to the JCL) and rerun the job.
40-02	The ST command failed because an ML entry is required	When a new table is defined in the TSR without any library DDNAMEs in the library list, the library on which the table is to be stored must be provided either by a DV or DW command before the STore command is issued or by specifying a valid library DDNAME in the ML-list in effect when the STore command is issued.
40-03	A blank or null DDNAME was used as a library list entry	A library DDNAME must be specified in the library list (ML-list) in effect when the command is being invoked.
40-04	The CA command failed because an ML entry is required	A CA command stores the alternate index definition on the first library in the library list (ML-list). A null or blank library list is not valid for this command.
40-05	The library list DDNAME entry is not allocated.	The current library list (ML-list) contains an entry that does not correspond to an allocated DDNAME. Correct the library list or add the DDNAME to the JCL and rerun.
41-00	The STORAGE-MODE-CODE (SMC) specified must be R or blank	Change the SMC value to “R” and resubmit the command. “P” for Page Table is not supported in V6.
42-00	The TBPARM sub parameter TB-FORMAT must be either 0 or A	Correct the value in the TB-FORMAT sub-parameter and resubmit the command. (See the *Programming Guide* for details on TBPARM sub-parameters.)
43-00	The estimated NUMBER-OF-ROWS value in the DT block is out of range	When a table is defined, the estimated number of rows times the row size cannot exceed 2G. It also cannot be negative.
44-00	The EXPANSION-FACTOR specified in the DT block must be from 1 to 999	Correct the value in this field and resubmit the command. Consult the *Programming Guide* for the usage of this parameter.
49-00	This command requires more parameters than were supplied.	Supply the minimum number of parameters required for this command. Consult the *Programming Guide* and resubmit the command.
50-00	The DIRTYPE specified for the LD command must be V, D, T, L, or blank.	Correct the DIRTYPE used by the LD command and resubmit the command. (Consult the *Programming Guide* for the usage of the DIRTYPE parameter)

Return Code	Text	Meaning / Instructions
51-00	HIGH/LOW DENSITY specified in DT block must be from 1 to 999.	Correct the DENSITY value being used and resubmit the command. Consult the *Programming Guide* for the usage of these parameters. Note that the specified HIGH-DENSITY and LOW-DENSITY parameters can be adjusted dynamically by the HASH-HI-DEN-LIM and HASH_LOW_DEN_LIM run-time options.
55-00	The SEARCH-METHOD must be either S, Q, B, C, or H	Correct the SEARCH-METHOD value being used and resubmit the command. Consult the *Programming Guide* for the usage of this parameter.
56-00	The SEARCH-METHOD is incompatible with ORGANIZATION specified	Correct the SEARCH-METHOD and/or the ORGANIZATION field ensuring they are compatible and resubmit the command. (Consult the *Programming Guide* for the relationship of these two parameters).
58-00	The requested module cannot be found in the load libraries searched.	Add the missing module to the load libraries accessed through STEPLIB, LPA or linklist.
60-00	The specified library is not suitable.	The dataset specified is not a valid tableBASE library.
60-01	The specified library's data definition must be DASD	Ensure that the library DDName is allocated to DASD, and resubmit the command.
60-02	The specified library's BLKSIZE must be at least 600.	Increase the library BLKSIZE to at least 600 and resubmit the job.
60-03	The specified library must contain at least 10 blocks.	Increase the number of blocks in the library; 10 is the minimum. Correct the allocation and resubmit the command.
60-04	TableBASE libraries are limited to 8,388,607 blocks.	Reduce the number of blocks in the library. The maximum for a BDAM library dataset is the number of blocks that can be contained in 65,535 tracks (approximately 4369 cyl). The maximum for a un-extended VSAM RRDS library dataset is the number of blocks that does not exceeed 4G bytes of data (approx 6145 cyl). Correct the allocation and resubmit the command.
60-05	The BLKSIZEs of the source and target libraries differ.	The process being invoked requires that the source and target libraries have the same blocksize. Correct the blocksizes and resubmit the command.
60-06	The size of the designated library is smaller than that of the source.	Reallocate the target library dataset to contain the same as or more blocks than the source dataset and rerun the job.
60-07	The specified library's BLKSIZE is not known	The library is not a valid tableBASE library. Ensure that the library has been successfully DEFINEd with either the DL command or the DK1TEXEC (TBEXEC) Batch Utility. Rerun this job after the library is successfully defined.
60-08	The library expansion is not complete. Recreate the library.	An I/O error was experienced performing the library expansion. Recreate the library and resubmit the job.
60-09	The target library cannot be a V5 Bridge library.	Library Expand failed. Ensure the target library is a V6 format library and resubmit the job.
61-00	The library status is invalid.	The tableBASE library DD is not allocated appropriately
61-01	The library DDNAME may not be concatenated with another dataset.	tableBASE Library DDNAME concatenation is not supported. Ensure each library has a separate DDNAME and resubmit the command.
61-02	The DSNAME is open with another DDNAME	Only 1 DDNAME is allowed per library DSNAME. Correct the JCL and resubmit the job.
61-03	The DL command or library expansion failed, the library was open	The library was already in use. Ensure the process has exclusive use of the library and resubmit the job.
61-04	The DL command or library expansion failed, DISP must be NEW or OLD	Correct the library disposition in the JCL and resubmit the job.

Return Code	Text	Meaning / Instructions
61-05	The DL command failed; a VSAM library was not defined as reusable.	Redefine the VSAM library as reusable and resubmit the job.
61-06	The designated library access method must be BDAM or VSAM.	Correct the DSORG of the library and resubmit the job.
61-07	The library DISP must be OLD or SHR for this command.	Correct the library disposition in the JCL and resubmit the job.
61-08	OPTCD=C and Z mutually exclusive on tableBASE library DD statement.	Correct the OPTCD DD statement and resubmit the job. See the *Administration Guide* for documentation on tableBASE use of the OPTCD subparameters C and Z.
61-09	The BDAM library must be RECFM=F or FB	Correct the BDAM library record format and resubmit the job.
61-10	The VSAM library definition must be RRDS (Relative Record dataset).	Correct the VSAM library definition and resubmit the job.
61-11	A write operation to the library was terminated, library read-only.	No updates are allowed to this library because it is defined with read-only access. In Batch jobs, LABEL=(,,,IN) on the DD statement defines a library dataset as read-only; in CICS, the library file definition is read-only. If updates to the library are to be allowed, change the JCL or CICS file definition. If no updates are allowed, inform the owners of the application to change their practices.
61-12	The specified library is RACF protected for this type of access.	The RACF/AUTHORITY does not allow the specified access to the library dataset. Determine whether to request RACF changes or inform the owners of the application to change their practices.
61-13	The specified file is either disabled or being closed	Access to this library is no longer possible under CICS because the file was disabled or closed. Determine whether the change in CICS file status should be reversed and resubmit the request.
62-00	Format of the specified library is not compatible with Version 6.	TableBASE V6 does not support tableBASE libraries that have not been converted from V5 or defined under V6. To determine library format run job DK1TLCHK. Ensure the library is in a compatible format and resubmit the job.
64-00	The INDEX parameter specified must be P, T or blank	Correct the index parameter and resubmit the command. (Consult the *Programming Guide* for the usage of the INDEX parameter)
68-00	The COUNT value is too small for the DU command to dump all rows	This Error Code may be OK if only a portion of the rows were to be dumped. If all the rows in the table were to be dumped, Increase the COUNT value to the total number of rows and resubmit the command. (Consult the *Programming Guide* for the parameters of the DUmp command).
70-00	TableBASE cleanup unable to complete due to CICS shutdown processing.	tableBASE's attempts to clean up control blocks in the CICS region at termination cannot complete due to CICS termination processing.
71-00	Lock is unavailable after waiting LOCKTIMERC seconds.	A tableBASE lock was not available for use after waiting LOCKTIMERC seconds. Resubmit the command. If a longer wait time for the resource is desired, update the TBOPT LOCKTIMERC value and resubmit the command.
72-00	The table is unavailable at this time; WAIT switch is off.	The table was in use at the time of the command. Retry the command.

Return Code	Text	Meaning / Instructions
72-01	The table is unavailable after waiting TABLEWAITRC seconds.	With the WAIT switch on, the table enqueue was unavailable after waiting TABLEWAITRC seconds. If the wait time is to be increased, update the TBOPT TABLEWAITRC value and resubmit the command. The table enqueue is an MVS enqueue with a major name of TBLBASE and a minor name starting with T followed by the table name and the library DSN. It is held while a table is open for write in a TSR.
73-00	An update command was issued for a table that was not open for write	An update command was issued for a table that was not open for write. This message will be returned in the following cases: 1) if the table is open for read in a local TSR, and this is not single-tasked batch (allows for backward compatibility) 2) if the table is open for read in a VTS-TSR 3) if the table is not open, implicit open is ON, and this is not single-tasked batch. In case 3, the table will not be left open after the command completes. The command being issued requires that the table be OPEN for WRITE (OW). For correct operation, issue an OW command for the table and retry the command.
74-00	The LOCK-LATCH specified is invalid	If a lock latch is set on a table in a TSR, any update command must match the lock latch value. Correct the LOCK-LATCH value and resubmit the command. If the lock-latch is not known, the tableBASE Administrator can perform the required command overriding the lock-latch with the Master Password.
75-00	Dynamic allocation (AL) or un-allocation (UL) failed	tableBASE command AL (Allocate library) or UL (Un-allocate library) was not successful.
75-01	TBTSLIB not allowed as parameter in AL command.	TBTSLIB is obsolete in V6. It cannot be dynamically allocated with the AL command.
75-02	DDNAME specified in AL command already in use.	Ensure the DDNAME is not already allocated and resubmit the command.
75-03	DSNAME specified in AL command not catalogued.	Ensure the DSNAME for the DDNAME is cataloged and resubmit the command. The AL command only supports catalogued library datasets
75-04	DSN enqueue conflict with another region for AL command.	Determine which region has the SYSDSN enqueue for this table and resolve the conflict. Then reissue the AL command with the appropriate SHR parameter. (Consult the *Programming Guide* for the usage of the AL command)
75-05	Error in DDNAME specified in AL command.	Correct the DDNAME syntax and resubmit the command
75-06	Syntax error in DSNAME specified in AL command.	Correct the DSNAME syntax and resubmit the command
75-07	SVC99 failure during execution of AL command.	An MVS error was generated. Contact your MVS systems programmer to determine why the AL command failed.
75-08	DDNAME specified by UL command must be allocated by an AL command.	The UL (Un-allocate) command failed because tableBASE was not used to allocate the DDNAME. Deallocate the DDNAME (TSO FREE) and use the tableBASE AL command to allocate the library dataset.
75-09	DDNAME specified by UL command not known or library not open.	Ensure the DDNAME is allocated.
75-10	SVC99 failure during execution of UL command.	An MVS error was generated. Contact your MVS systems programmer to determine why the UL command failed.
76-00	OI (Open Indirect) failed: primary table not sequential or KLOC not 9.	Ensure the Open Indirect primary table has sequential organization and the key location is 9. Resubmit the command. (See the *Programming Guide* for details on Open Indirect, or the SI command for more details.)

Return Code	Text	Meaning / Instructions
77-00	An alternate index is defined for a table that is not TYPE=P.	Ensure the data table is defined as a pointer table (TYPE=P) and resubmit the command
78-00	The AL command failed, the SHARE-STATUS parameter must be S or O	Correct the SHARE-STATUS sub-parameter of the LIBRARY-ALLOC parameter. (See the *Programming Guide* for the usage of the AL command.)
80-00	Data table is not open; the use of an alternate index not allowed.	The IA command requires the data table be opened first. The OR command will dynamically open a data table.
80-01	Data table must be opened before IA (Invoke Alternate) is issued.	Open the data table before issuing the IA command.
80-02	IA (Invoke Alternate Index) may not reference a linked VTS table.	Both the alternate and the data table must be opened in the VTS-TSR or opened in the local TSR for IA to succeed. The base table cannot be a linked VTS table.
80-03	Open/Invoke Alternate Index must reference a data table.	Ensure the IA command includes the BASE-TABLE-NAME parameter and resubmit the command.
80-04	Alternate Index cannot be opened as a linked VTS table.	The IA command specified an alternate index name which was found in a VTS-TSR. This is not supported. Either specify the complete alternate index definition in the IA command parameters, remove the VTS-TSR from the library list (ML-list), or close the alternate index table in the VTS-TSR and resubmit the command.
81-00	The specified Alternate Index definition is not found	It the complete alternate index definition is not specified in the IA command parameters, the alternate index definition must be found in a library in the current library list (ML-list). Correct this error and resubmit the command.
83-00	Data table INDEX sub-parameter of the DT block must be defined as P	If the base table is specified in the IA command, it must be defined as a pointer table (TYPE=P). Redefine the base table.
85-00	The command is invalid for an Alternate Index	The command being invoked is not supported for use with an alternate-index. Correct the error and resubmit the command.
85-01	An alternate index may not have the same name as the data table.	Correct the alternate-index parameters ensuring that the alternate index and base table names are different and resubmit the command.
85-02	DV and DW commands are not supported for Alternate index tables	The DV and DW commands are only supported for base tables since they apply to ensuing STore commands. An alternate index is never stored, only the base table is.
86-00	Data table of an Alternate Index may not be a linked VTS table.	The table failed to open because the alternate-index is in the local TSR, and the data-table is linked to a table in a VTS-TSR. The alternate index must be opened in the same TSR as the data-table.
87-00	The KEY-COUNT must be 1 for the definition of the Alternate Index.	Only a value of 1 for the key-count is supported when defining an alternate-index. Correct the key-count value and resubmit the command.
89-00	An invalid parameter was encountered in the TBOPT file.	During initialization of tableBASE in the region, an error was discovered in the tableBASE runtime options in the TBOPT dataset. Correct the TBOPT parameter(s) and resubmit the job.
90-00	Insufficient storage available in region: increase region size.	There is insufficient main storage available for tableBASE control blocks. Increase the executing region size and resubmit the job (modify the 'REGION=' parameter for the jobstep).
91-00	I/O Error	Determine the reason for the I/O error from the JESMSGLG. Correct the problem and resubmit the command.
91-01	I/O Error while attempting to read/write to a tableBASE library	Determine the reason for the I/O access error to the tableBASE library. Correct the problem and resubmit the command.

Return Code	Text	Meaning / Instructions
91-02	I/O Error while attempting to read from the TBOPT file	Determine the reason for the error trying to read from the TBOPT file. Once resolved, resubmit the job.
91-03	Error in I/O subtask in multitasking environment.	The subtask was unable to complete due to an I/O error when TBOPT MULTITASKING=Y. Correct the problem and resubmit the process.
92-00	Insufficient Table Space Region (TSR) size	The TSR is too small for the tables being opened. Either enlarge the TSR or close unnecessary tables in the TSR to free up the required TSR space.
92-01	The space request is bigger than the TSR	The table to be opened is too large for the current TSR. The TSR size will need to be increased to accommodate the table to be opened.
92-02	Insufficient free space in the TSR: close tables or enlarge TSR.	The TSR is full and no longer has enough free space available to allow the table to be opened, defined, expanded or changed. Either enlarge the TSR appropriately or close unnecessary tables to free up the required TSR space.
92-03	There is insufficient free contiguous space in the TSR	The TSR is full. There is no longer any contiguous memory left to accommodate the index for the table being opened, defined, expanded or changed. Either enlarge the TSR or close unnecessary tables to free up required TSR space.
92-04	There is insufficient space in the TSR to initiate the TCE	The TSR is full. There is no room to accommodate the tableBASE control block created when opening a table into the TSR. Either enlarge the TSR or close unnecessary tables to free up the required TSR space.
92-05	The specified TSR size is invalid	tableBASE cannot initialize because the value set for the TSR size is not valid. Correct the value and resubmit the job.
93-00	The RF command did not complete	Correct the error and resubmit the command.
93-01	The table must be open for read in order to be refreshed	Open the table for read (OR) or ReLease the table before submitting the RF command.
93-02	The data table or alternates were closed since the table was opened	There is a change in either the alternate or the data-table that is preventing successful completion of the RF command. CL the table and re-open it from the desired library to get the most recent version of the table into the TSR.
93-03	The ROWSIZE changed since the table was opened	RF failed because the row size for the table on the library was different compared to the table opened in the TSR. CL and reopen the table to get the most recent version of the table into the TSR.
93-04	The KEYSIZE changed since the table was opened	RF failed because the key size for the table on the library was different compared to the table opened in the TSR. CL and reopen the table to get the most recent version of the table into the TSR.
93-05	The KEYLOC changed since the table was opened	RF failed because the key-location for the table on the library was different compared to the table opened in the TSR. CL and reopen the table to get the most recent version of the table into the TSR.
93-06	The ORGANIZATION changed since the table was opened	RF failed because the organization for the table on the library was different compared to the table opened in the TSR. CL and reopen the table to get the most recent version of the table into the TSR.
93-07	The SEARCH METHOD changed since the table was opened	RF failed because the Search-Method for the table on the library was different compared to the table opened in the TSR. CL and reopen the table to get the most recent version of the table into the TSR.

Return Code	Text	Meaning / Instructions
95-00	The transaction was terminated during tableBASE processing.	tableBASE cleanup was invoked because a transaction was terminated while tableBASE was processing. The subcodes document what tableBASE process was executing when the transaction was terminated. This is usually related to a transaction timeout or cancellation.
95-01	The transaction thread abnormally terminated	See 95-00 description. Send the documentation (JESMSGLG, TBDUMP, any other dumps) to tableBASE support.
95-02	tableBASE found an invalid row (item) area address	See 95-00 description. This can be caused by an error in the input parameters provided.
95-03	tableBASE found an invalid row address in the Table Space Region (TSR)	See 95-00 description. Send the documentation (JESMSGLG, TBDUMP, any other dumps) to tableBASE support.
95-04	tableBASE abended during an index resorting process.	See 95-00 description.
95-05	tableBASE abended during an index expansion process.	See 95-00 description.
95-06	tableBASE abended during an index entry insertion process.	See 95-00 description.
95-07	tableBASE abended during an index entry deletion process.	See 95-00 description.
95-08	tableBASE abended during an open command.	See 95-00 description
95-09	tableBASE abended during a close command.	See 95-00 description.
97-00	The ML or LL parameter list is invalid	Correct the ML or LL parameter list and resubmit the command.
97-05	An ML list must not contain a reference to TBTSLIB.	Remove the reference to TBTSLIB in the ML-list and resubmit the command.
97-08	An ML list for a VTS agent must not contain a reference to a VTS-TSR.	Only tableBASE libraries can be used in an ML-list in VTS. Remove the VTSname from the ML-list and resubmit the command.
98-00	An internal error occurred in tableBASE Shutdown processing.	tableBASE termination (XX command) problems.
98-01	The XX command is being processed on a subordinate task.	When multi-tasking is ON, the XX command must be performed by the persistent (mother) task. This command will fail if it is submitted by sub-tasks invoked by the persistent (mother) task. Send the documentation (JESMSGLG, TBDUMP, any other dumps) to tableBASE support.
98-02	tableBASE shutdown process did not complete.	If multi-tasking and a sub-task submitted the XX command, other tasks will first issue a warning, then they will abend. This is the warning. Send the documentation (JESMSGLG, TBDUMP, any other dumps) to tableBASE support.
98-03	The subtask that shutdown tableBASE has tried to invoke tableBASE.	If multi-tasking and a sub-task submitted the XX command, no tasks are allowed to issue tableBASE commands. Send the documentation (JESMSGLG, TBDUMP, any other dumps) to tableBASE support.
1072-00	VTS access failed because the specified VTS-TSR is not available.	VTS access failed because the specified VTS-TSR is not available. Ensure that the VTS Agent job which creates the specified VTS-TSR name is running.
1072-01	VTS access failed because the specified PC Server was not running.	Access to VTS-TSRs depends on a PC Server at the same or higher level than the application is running on the same LPAR.

tableBASE messages

Messages issued from the tableBASE components, such as the VTS Agent, CICS, batch and others, vary from function to function, and are listed in the table below. These messages can usually be viewed on the JESMSGLG. Any Severe Errors (S) messages should be referred to tableBASE Customer Support. Messages are in the format DK1nnnnnA where DK1 is the prefix, followed by a five digit numeric code and an error code letter. The error code letter indicates the type and severity of the error message:

E—Error (usually indicates user error)
I—Information
W—Warning
A—Action (user action is required)
S—Severe (no further processing is possible)

Table E-2: Messages and error codes for VTS Agent, CICS, batch, etc.

Msg Code	Text	Meaning / Instructions
DK100049S	NO CMDAREA IN CALL TO TBLBASE. ABENDING.	There was no command area supplied. Ensure that a command area is supplied for the tableBASE call.
DK100200S	Initialization failed	
DK100201E	PC Service unavailable; Required for VTS-TSR use	
DK100201I	tableBASE PC SERVER UNAVAILABLE	Information only.
DK100202I	tableBASE <version> in initializing for <system license info>	Information only.
DK100203I	tableBASE <type> license expires on <date>	Information only.
DK100207E	DK1TX071 not loaded; System Exits disabled	
DK100207E	DK1TX072 not loaded; User Exits disabled	
DK100210E	Module DK1TNUCL not loaded	
DK100212E	MULTITASKING requested or VTS Agent; see message DK100201E	
DK100213E	Error in tableBASE PC server	
DK100214E	CMA GETMAIN failed	
DK100221E	I/O ERROR:	
DK100222E	QSAM ERROR: FUNCTION=xxxxxxxx,FILE=ffffffff	
DK100224E	I/O PROBLEM: FILE=xxxxxxxx FUNC=ffffffff RESP=dddd RESP2=dddd	
DK100226I	Please change DD TBOPTV to TBOPT	Information only.
DK100227W	JOB <jobname> IN vvvvvvvv WAITING FOR nnnn SECONDS FOR TABLE <tablename>	
DK100228W	JOB <jobname> IN <VTSname> WAITING FOR <number> SECONDS FOR TABLE <tablename>	Job is waiting for VTS-TSR table enqueue--not received yet.
DK100230I	TBOPT Processing	Information only.
DK100231I	keyword=value.....	Information only.
DK100232E	Invalid value	
DK100233E	Missing keyword	

Msg Code	Text	Meaning / Instructions
DK100234E	Invalid keyword	
DK100235E	Value exceeds maximum allowed	
DK100236E	Keyword not allowed in this environment	
DK100237E	Missing "="	
DK100238E	Invalid quoted string	
DK100239E	Value field too wide	
DK100240E	Value less than minimum allowed	
DK100241E	Missing value	
DK100242E	Keyword (or alias) previously specified	
DK100243E	Option is not supported in this version	Your organization is not authorized to use the optional product; please contact Technical Support.
DK100244I	MAXNMTAB set to nnnnnn	Information only.
DK100245W	VTSONLY not valid with LIB entries	
DK100246W	Invalid VTS prefix for LIBnn=xxxxxxxx	
DK100247W	Empty value: LIBxx - LIByy	
DK100248W	LIBxx exceeds max allowed mm	
DK100249W	MULTOPNX=N is not valid	
DK100250I	TBOPT Parameters: (*= If default)	Information only.
DK100251I	keypwrd=value.....	Information only.
DK100252I	<<<<<<< TBOPT Done >>>>>>	Information only.
DK100253I	<message>	Information only.
DK100260I	DK1TX066 Exit Manager Initializing	Information only.
DK100262W	Exit xxxxxxxx not loaded	
DK100263I	Exit Manager Shutting down; Exits active	Information only.
DK100264I	DK1TX066 Exit Manager Shutdown complete	
DK100265W	DK1TC073 NOT LOADED	
DK100270E	Insufficient storage to process strobe data; strobe suppressed.	
DK100271E	Error on CICS WRITE JOURNALNUM; Strobe logging suppressed	
DK100272E	WRITE JOURNALNUM RESP=1234, RESP2=1234	
DK100273E	Dynalloc of strobe report failed. Strobe suppressed.	
DK100275E	Module DK1TRSTA not found.	
DK100277I	>> ESTAI -NO CMA	Information only.
DK100280E	QSAM ERROR: FUNCTION=xxxxxxxx,FILE=ffffffff	
DK100281E	STROBE OUTPUT SUPPRESSED	
DK100282E	I/O ERROR:	
DK100292E	DEBUG TRACE requested but not started	
DK100295A	Specify character for PC server Named Token	
DK100299	DK1T1153 PRE-XDC	
DK100299	DK1T1153 XDC ENABLED	

Msg Code	Text	Meaning / Instructions
DK100300S	SYSTEM FAILURE: xxxxxxxx Code=cccc/ hhhhhhhh, Reason=rrrr/hhhhhhhh	
DK100300S	Call from ???????? Offset ????	
DK100300S	R0-R7 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx ...	
DK100300S	R8-R15 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx ...	
DK100301E	LOGIC ERROR DUMP TAKEN. DUMPCODE IS LGIC. DUMPID IS 12345678	
DK100301S	LOGIC ERROR; DUMP TAKEN TO TBDUMP	
DK100302S	System ENQ limit exceeded. Task abended with tableBASE	
DK100310W	TABLEBASE TSR nn% ALLOCATED	The series of space management algorithms used in prior levels of V6 was changed in level8. It now only issues a message when the TSR usage rises above a threshold and does not indicate what algorithm is being used. Setting the TSR_WARNING_FREQ = 0 will cause all the messages to be displayed once the TSR space usage threshold is crossed from below the threshold to above it or exactly equal to it. The default FREQ setting is 1 (i.e. only display one message per second, at the most). Level 8 will issue this message anytime the TSR usage rises above the value in TSR_WARNING_PCT. (The Default TSR_WARNING_PCT is 85.) Level 8 warnings start when the usage is exactly the TSR_WARNING_PCT value or higher. For more details refer to the TBOPT parameter settings in the *tableBASE Installation Guide*
DK100330I	DDNAME xxxxxxxx IS NOT A TABLEBASE Version 6 LIBRARY	Information only.
DK100331W	LIBRARY FREE SPACE MAP INCONSISTENT WITH LEFT BLOCK COUNT	
DK100340W	Old form ML mixed with use of extended ML	
DK100341W	CK, CO, CP COMMANDS NOT SUPPORTED YET	
DK100400E	tableBASE User Error Abend G200, Reason=0999/03E7, tableBASE Initializing	tableBASE license has expired. Call DataKinetics Ltd. at (613)523-5500 to renew.
DK100411E	TSR Create failed: Failed by site IEFUSI	
DK100412E	TSR Create failed: MVS resource shortage	
DK100413W	SNAP SUPPRESSED; GETMEM SHORTAGE	
DK100470E	BDAM ERROR: FUNCTION=xxxxxxxx,FILE=ffffffff	
DK100471E	VSAM ERROR: FUNCTION=xxxxxxxx,REG15=rrrrr,ERROR CODE=xxxxx,FILE=ffffffff	
DK100472E	I/O ERROR: followed by text of I/O error	
DK100475E	xxAM ERROR: FUNCTION=xxxxxxxx,FILE=ffffffff	
DK100476E	VSAM ERROR: FUNCTION=xxxxxxxx,REG15=rrrrr,ERROR CODE=xxxxx,FILE=ffffffff	

Msg Code	Text	Meaning / Instructions
DK100477E	I/O ERROR: followed by text of I/O error	
DK100490E	TBL BASE ERROR DEBUG WTD, REASON=xxxx, DEBUG DISABLED	
DK100491I	ROTB EOTX	Information only.
DK100492I	ROTBEOT: ABND=123/123	Information only.
DK100493I	ROTB NORMAL END	Information only.
DK100494E	PERSISTENT TASK ENDED BEFORE END OF STEP	
DK100495I	IN ROTBVTSD	Information only.
DK100500E	tableBASE ERROR ABEND xxxx, Command=xx,xxxxxxxx, Reason=dddd/hhhh	
DK100501S	EXEC CICS FAILURE: xxxxxxxx Resp=cccc/ hhhhhhhh, Resp2=cccc/hhhhhhhh	
DK100511E	TSR Create failed: Failed by site IEFUSI	
DK100512E	TSR Create failed: MVS resource shortage	
DK100513W	SNAP suppressed; GETMEM shortage	
DK100530I	No storage for QTCWA; QCT function not used.	Information only.
DK100531I	Module DK1TRSTA not found; QCT function not used.	Information only.
DK100550I	tableBASE 6.0 - Initializing	Information only.
DK100551I	tableBASE 6.0 - Initialized	Information only.
DK100552E	tableBASE 6.0 - Initialization Error	
DK100553E	tableBASE 6.0 - Initialization Failed	
DK100554I	tableBASE 6.0 - Starting tableBASE	Information only.
DK100555I	tableBASE 6.0 - tableBASE started	Information only.
DK100556I	tableBASE 6.0 - Deactivating	Information only.
DK100557I	tableBASE 6.0 - Deactivated	Information only.
DK100558E	tableBASE 6.0 - Deactivation Error	
DK100559I	tableBASE 6.0 - Deactivation Failed	
DK100560I	tableBASE 6.0 - Restarting	Information only.
DK100561I	tableBASE 6.0 - Verifying	Information only.
DK100562I	tableBASE 6.0 - Verification completed	Information only.
DK100563I	tableBASE 6.0 - Error Info: Func(function12345678/subfuncn)	
DK100564I	DK1TCIN used in PLT takes one of INIT (default) or TBINIT	Information only.
DK100565I	Successful xxxx INIT must be done before xxxx TBCALL	Information only.
DK100566I	Verification failure	
DK100570I	xxxx takes one of INIT, TERM, RESET, TBINIT, TBTERM,TBRESET, TBCALL, VER, HELP	Information only.
DK100571I	Phase in of xxxxxxxx not allowed by CICS; current in-storage copy used.	Information only.
DK100572I	12345678 LP=xxxxxxxx EP=xxxxxxxx AM=xx RC=xxxxxxxx UC=xxxxxxxx	Information only.
DK100573I	Module 12345678 not loaded	Information only.

Msg Code	Text	Meaning / Instructions
DK100574I	Module 12345678 not found.	Information only.
DK100575I	EX=xxxxxxxx EN=xxxxxxxx STST=xxxx SHST=xxxx TAST=xxxx TALN=xxxxx GALN=xxxxx	Information only.
DK100576I	xxxxxxxxxxxxxxxx/xxxxxxxx OK	Information only.
DK100577I	xxxxxxxx LP=xxxxxxxx EP=xxxxxxxx LEN=xxxxxxxx AM=xxx KEY=x	Information only.
DK100590E	TBL BASE ERROR DEBUG WTD, REASON=xxxx, DEBUG DISABLED	
DK100595I	Trans error dump taken. Dumpcode is DKL1. Dumpid is xxxxxxxx	Information only.
DK100596E	CICS Assign failed (Resp/Resp2=nnnnn/nnnnn); dump suppressed	
DK100597E	CICS DUMP TRANSACTION failed (Resp/ Resp2=nnnnn/nnnnn). Dump for userid abcd1234 not taken.	
DK100600I	tableBASE VTS xxxxxx initialized	Information only.
DK100601E	GCA cannot be located; VTS stopping	The PC Server is not running. It must be running before any VTS-TSRs can be brought up (for systems NOT running the optional tableBASE Process Manager).
DK100602E	VTS Agent incompatible with GCA structure	
DK100603E	Duplicate VTS attempting to start; auto-stopping	
DK100604E	Program x2345678 is not APF authorized	
DK100605E	Unable to locate tableBASE GCA	The TPVM (VTS Manager) COMPAT is not running. It must be running before any VTS-TSRs can be brought up (for systems running the optional tableBASE Process Manager).
DK100610E	Error in xxxxxxxx macro; RC/REASON=xxx/yyyy	
DK100630I	Console interface is available	
DK100631E	Invalid command; command ignored	
DK100632W	'REFRESH not supported in tableBASE V6 VTS	
DK100633I	SHUTDOWN	Information only.
DK100634I	STOP command received	Information only.
DK100690A	VTS Agent waits, non-swappable; reply D to terminate	
DK100691A	Specify character for PC server Named Token	
DK100800I	tableBASE PC server V6 available	Information only.
DK100802E	PC server xxxxxxxx terminating; see message DK100803I for reason.	

Msg Code	Text	Meaning / Instructions
DK100803I	ERROR IN PARM FIELD ERROR IN GCA STRUCTURE ANOTHER tableBASE PC SERVER IS RUNNING LIBRARY xxxxxxxx IS NOT APF xxxxxxxx MODULE xxxxxxxx IS NOT APF BOTH STEPLIB AND SYSLIB ARE MISSING MODULE xxxxxxxx NOT FOUND IN ANY OF LPA, STEPLIB, OR SYSLIB LIBRARY xxxxxxxx MUST NOT BE CONCATENATED BLDL FOUND MODULE xxxxxxxx IN OTHER THAN STEPLIB/ SYSLIB OPEN xxxxxxxx FAILED CLOSE xxxxxxxx FAILED ERROR IN MACRO xxxxxxxx ; RC/REASON=xxx/xxxx PC SERVER NOT COMPATIBLE WITH GCA STRUCTURE	Information only.
DK100810A	PC server: Reply Q to terminate PC server	
DK100820A	PC server: Specify character for test Named Token	
DK100821I	Invalid command; ignored: xxxxxxxx	Information only.
DK100822I	STOP command received	Information only.
DK100830I	PC server: Utility functions -	Information only.
DK100830I	Purge GCA, Snap GCA, snap GCA Header, or Exit options.	Information only.
DK100831A	PC server: Specify p, s, h, or e.	
DK100832I	PC server: Exit options - U for Utilities, R to Resume PC server, or E to Exit PC server.	Information only.
DK100833A	PC server: Specify u, r, or e.	
DK100834I	PC server not compatible with GCA.	
DK100834I	Do you want to do a partial purge, i.e., NT, GCA, not GCAXs, PC code?	
DK100835A	PC server: Reply y for partial purge, n to bypass	
DK100840I	GCA does not exist.	
DK100841I	Negative GCA extension link.	
DK100842I	PC server not compatible with GCA; unformatted GCA dumped.	
DK100843I	GCA length value not valid; GCA header dumped.	
DK100844W	Customer Anchor Table DKL slot invalid.	
DK100845I	Allocate of DDname for snap dump failed.	
DK100846I	Open of DDname for snap dump failed.	
DK100847I	DDname specified for snap dump is in use.	Information only.
DK100848I	DDname specified for snap dump is invalid.	
DK100850I	DELETING NT	Information only.
DK100851I	NT DELETED	Information only.
DK100852I	FREEING PC CODE	Information only.

Msg Code	Text	Meaning / Instructions
DK100853I	PC CODE FREED	Information only.
DK100854I	FREEING GCA	Information only.
DK100855I	FREEING GCAX	Information only.
DK100856I	GCA PURGED	Information only.
DK100891I	RESXIT	Information only.
DK100990S	Incompatible ROOT and NUCLEUS versions	
DK100991E	Parms module not loaded	
DK100993E	Parms module corrupted	

TBEXEC error messages

Error and audit messages generated by the batch utility program TBEXEC are identified and described in alphabetical order in the following table.

Table E-3: tableBASE batch utility messages

Message	Text	Meaning/Instructions
Data table index must be defined as P	TBEXEC attempted to invoke an Alternate Index of a table that is not defined to be type P (indexed or pointer).	
Change definition failed	The attempt to change the definition of the table has failed for the reasons noted above the message.	
Change is not for Alternate Index	The CHANGE ALT= command specified a table that is not an Alternate Index. Use the CHANGE TBL= version of the command.	
Change key size/location failed	The attempt to change the key size and/or key location has failed for reasons noted in preceding messages or to the right of this message.	
Change maxgen failed	The attempt to change the number of generations to be kept has failed for reasons noted in preceding messages or to the right of this message.	
Change not made for reasons above	None of the changes in the above change were performed. The reason for the rejection of this command is noted above the message.	
Change successful	The table definition has been changed as requested.	Information only.
CMD requires specified table to be open	This command requires the specified table to be open. This error should not occur when using TBEXEC.	
Command is invalid for an Alternate Index	The command cannot be performed on an Alternate Index.	
Command rejected for reason above	The preceding TBEXEC Command was rejected for the reasons noted above the message.	
Command requires more parms than given	TBEXEC called tableBASE incorrectly.	
Copy complete	The requested operation completed successfully.	
Copy complete except where indicated	The requested copy operation was successfully completed except as noted above the message.	
Copy failed	The requested operation failed. Diagnostic messages are issued to give the reason(s) for the failure.	
Command is invalid	The specified command is not a TBEXEC command.	
Copy of table (all generations) complete	The requested copy operation of all generations of a table was successfully completed.	
Create Alternate Index definition failed	The attempt to create an Alternate Index definition failed for the reasons noted in preceding messages or to the right of this message.	
Create Alternate Index definition successful	The Alternate Index definition has been created successfully.	Information only.
Defaults have been set	The defaults entered have been set.	
Delete failed	The requested delete operation has failed for reasons noted in preceding messages or to the right of this message.	
Delimiter is invalid, = was expected	The next symbol after a keyword must be an equal sign.	
Density parameter must be from 1-999	The density must be numeric and in the range 1-999.	
Destination library too small	The destination library is too small to hold the table(s) to be copied.	

Message	Text	Meaning/Instructions
Directory is empty	The directory on the specified library is empty.	
DT command (maxgen parm) must be 1-9	The maximum generations to be kept must be numeric in the range 1-9 inclusive.	
Duplicate keyword for this command	The same keyword was specified twice for a command. Fix the command sequence and try again.	Fix the command sequence and try again.
End of data - tableBASE utility ended	End of data has been reached on CNTLCARD, the TBEXEC input file.	Information only.
Error In table definition	An error has been detected in the table definition created from the parameters supplied. Refer to the TBCALL Error Code for the reason.	
Estimated number-of-rows is out of range	The estimated number of rows is non-numeric, or is too large.	
Expand library failed	The requested operation failed. Diagnostic messages are issued to give the reason(s) for the failure.	
Expand library successful	The requested operation completed successfully.	Information only.
Expansion factor must be from 1 To 999	The expansion factor must be numeric and in the range 1-999.	
Field is greater than 8 characters	All of the keywords and keyword values have a maximum length of eight characters.	Ensure field is 8 characters or less.
Format of library incompatible with Version 6	The library format needs to be converted to be compatible with Version 6. Please see the tableBASE Installation Guide, Appendix B.	
Generation number specified is invalid	The generation must be numeric and in the range 0-255.	
Generation requested has been copied	The COPY operation has been completed successfully. A new table (generation number 1) has been added to the target library.	Information only.
Index parameter must be P, T or blank	The Index field must be T for true tables, P for indexed (Pointer) tables or blank. In previous releases, tableBASE allowed two types of tables: Pointer and True. In Version 6, the concept of True tables still exists, however they are treated within tableBASE as Pointer tables as all memory is now in segmented memory that requires Indexes. The True table Indexes will be transparent to the application program.	
Initialization successful	The new tableBASE library has been initialized successfully.	Information only.
Insufficient local tableSPACE region	The TSR is not large enough to contain the table and tableBASE internal tables. tableBASE uses a small part of the TSR for its own internal tables.	
Insufficient space on new library	There is not enough space on the target library for the tables being copied.	
INV storage-mode-code: Must be R, Blank	The storage mode code (SMC) must be R or blank.	
Key size invalid: must be1-256	The key size must be a number from 1-256, inclusive.	
Keyword and value are incompatible	If the keyword requires a numeric value, the value specified with it must be numeric. Alternatively, the ALLGEN keyword must be followed by = YES.	
Keyword invalid for this command	The keyword specified was not recognized.	
Keyword is incomplete	The keyword has not been followed by an equal sign (=)and a value.	
Keyword is invalid	The keyword specified is not used with this command.	
KLOC invalid: must be 1 to row-size	The key location must be numeric in the range 1 to the row size, inclusive.	

Message	Text	Meaning/Instructions
Library is not empty	The target or destination library of an EXPAND LIBRARY request contains at least one table. The target library must be empty for an EXPAND operation.	
Library not initialized	For the reason cited to the right of this message, the library was not initialized.	
Load failed	The requested operation failed. Diagnostic messages are issued to give the reason(s) for the failure.	

Mthd/org parameters are incompatible	This combination of search method and organization will not work. Valid combinations are:
	`Organization  Method`
	`------------  ------`
	`R, U          S`
	`S, D          S, B, C`
	`H             H`	

New generation loaded successfully	A new generation of this table has been created from the contents of the FROM dataset.	Information only.
Newlib DDNAME not Assigned -- check JCL	The requested initiation of the DDNAME specified by the NEWLIB parameter could not be performed since the DDNAME is not defined in the JCL.	Alter JCL as required.
nnnnn/mmmmm tables exported successfully	nnnnn tables were exported from the tableBASE library successfully. mmmmm tables were requested to be exported.	Information only.
nnnnn/mmmmm tables imported successfully	nnnnn tables were imported to the tableBASE library successfully. mmmmm tables were requested to be imported.	Information only.
No changes specified	A change command did not identify any fields to be changed.	Identify fields as required.
Not copied, new name same as old name	A COPY table request specified the same value for NEWNAME as for TBL.	
Not enough space on library	There is not enough space on the tableBASE library for the table being defined, copied, loaded, or expanded.	
Old Alternate Index definition deleted	During a copy of an Alternate Index definition, the old definition was deleted from the target library, but the new definition could not be copied from the source library to the target library.	
On dest lib - insufficient space for copy	The copy operation has been requested to a destination library with insufficient space on it to receive all generations of the tables on the FROM library.	
Org parm invalid: must be R, U, S, D or H	The Organization must be R, U, S, D, or H.	
Paged tables are not supported	Version 6 no longer supports paged tables.	
Print request completed successfully	The requested print operation has finished with no errors. The listing can be found in TBRPT.	Information only.
Rename failed	The requested operation failed. Diagnostic messages are issued to give the reason(s) for the failure.	
Rename successful	The requested rename operation was successfully performed.	Information only.
Requested generation has been cleared	A new generation of this table has been created using the definition of the generation specified. This new generation contains no items.	Information only.
Requested generation has been deleted	The generation of the table specified has been deleted.	Information only.
Requested generation(s) deleted	The requested operation completed successfully.	Information only.
Required keyword missing for this command	One of the required keywords for this command was not contained in the command sequence.	Correct command sequence as required.

Message	Text	Meaning/Instructions
RN fails. new name Already Exists	The table was not renamed because a table with the new name already exists on the library.	
Row size invalid: must be 1-32767	The row size must be numeric in the range 1-32767 inclusive.	

Search-method incompatible with organization	The search method must be S, Q, B, C, or H and must match the organization. Valid combinations are:
	`Organization  Method`
	`------------  ------`
	`R, U  S`
	`S, D  S, B, C`
	`H  H`	

Semi-colon is missing from command	The command and its keywords must be followed by a semicolon (;) to indicate the end of the keyword list for the command sequence.	
Source library is empty	The source library in a copy operation contains no tables.	
Table already exists on library	The table could not be defined on, or copied to, the new library because a table of the same name already exists on the target library.	
tableBASE error detected	The requested operation failed. Diagnostic messages are issued to give the reason(s) for the failure.	
tableBASE internal program error - xxxx	The requested operation failed. Diagnostic messages are issued to give the reason(s) for the failure.	
Table definition successful	The table was defined successfully and generation 1 was stored on the library.	
Table is not closed	This error should not occur when using TBEXEC.	Contact Technical Support.
Table is not found	The table was not found on the given library.	
Table name specified is invalid	A valid tableBASE table name is a string of 8 bytes that are not all blanks, all low values, all high values, or :TMPNAME.	
Table opened for read cannot be stored	A table has been opened for read-only access and a store or write command has been issued.	
Table type is invalid	The table type must be F, V, A, or X.	
Table unavailable. No wait in effect	TBEXEC could not store the table because it is locked by another application. The NO WAIT parameter is in effect.	
Table unloaded successfully	The specified generation of the specified table has been unloaded to the TO dataset.	Information only.
Table updated as requested	The requested update operations on the specified table were performed successfully.	Information only.
Table updated successfully	The requested operation completed successfully.	Information only.
Table xxxxxxxx could not be opened	For reasons noted in preceding messages or to the right of this message, the table xxxxxxxx could not be opened. Since the table must be opened to perform this TBEXEC command, the TBEXEC command could not be successfully completed.	
Table xxxxxxxx export failed	The export of table xxxxxxxx failed for the reasons given in other messages.	
Table xxxxxxxx import failed	The import of table xxxxxxxx failed for the reasons given in other messages.	
Table xxxxxxxx import replace failed	The import of table xxxxxxxx failed for the reasons given in other messages.	
Table xxxxxxxx sort error	The sort required for this operation failed for the reason noted.	
Table xxxxxxxx was not stored	The table xxxxxxxx was not stored for the reasons noted in preceding messages or to the right of this message.	
tablebase error detected	A TBLBASE error was detected in the processing of this command. The explanation of the error is found to the right of this message.	

Message	Text	Meaning/Instructions
The count specified is invalid	The count is less than one, or greater than the number of rows in the table.	
The key will not fit within the row	The end of the key cannot exceed the end of the row.	
The library DDNAME does not exist	The LIB=DDNAME is not in the JCL.	Alter JCL as required.
The library status is invalid	tableBASE library disposition must be NEW for new libraries, and SHR or OLD for existing libraries.	
The password supplied is invalid	An incorrect password has been specified.	
The specified command is invalid	The specified command is not a valid TBEXEC command.	
The specified library is not suitable	There are a number of conditions that can create this error. See tableBASE error codes.	
Write password is missing or incorrect	The operation requires access to a write-protected table, but the write password has not been specified or specified incorrectly.	
xxxxxxxx field is greater than 8 characters	All keywords and keyword values must be eight characters or fewer.	

Table E-4:

tableBASE Process Manager messages

The following table contains information on the tableBASE Process Manager error messages that can be encountered. Note that the *Error Code* is, in all cases, in the format DK1PnnnnnX, where X is a letter code suffix, which indicates the type and severity of the error message:

E—Error (usually indicates user error)
I—Information
W—Warning
A—Action (user action is required)
S—Severe (no further processing is possible)

Table E-5: tableBASE Process Manager messages and error codes

Error Code	Text	Meaning / Instructions
DK1P02000E	RECOVERY PARMS NOT AVAILABLE - RECOVERY STOPPING	
DK1P02001I	TB 6.1 INITIALIZATION PROCESSING STARTING	Information only.
DK1P02002S	TPM START ENQ NOT AVAILABLE - TERMINATING	
DK1P02003I	P1063 TRMEXIT: TPM CB NOT FOUND OR NOT RUNNING	
DK1P02004S	INITIAL SET RECOVERY-ON FAILED - TERMINATING	
DK1P02006S	INITIAL SNAPX OPEN FAILED - TERMINATING	
DK1P02007S	INPUT PARM(S) NOT CORRECT - TERMINATING	
DK1P02008S	MESSAGE INIT REQUEST FAILED - TERMINATING	
DK1P02009I	TPM START PARM USED: <parm>	Information only.
DK1P02010S	DK1PMGR NOT AUTHORIZED - TERMINATING	Load library which contains DK1PMGR is not an authorized library. Make sure you are running from an authorized library.
DK1P02011I		Not used.
DK1P02012S	MODESET macro failed - terminating	
DK1P02014S	RESMGR macro failed - terminating	
DK1P02015I	Recovery detected /C (cancel) command	Information only.
DK1P02016S	Failure setting recovery on - terminating, routine = <routine name>, RC= <return code #>	
DK1P02017I	P1063 TRMEXIT: PROC ERROR - POSSIBLE JCL ERROR	
DK1P02018S	Failure setting recovery off - terminating, routine = <routine name>, RC = <return code #>	
DK1P02019I	P1063 TRMEXIT: POST COMPLETED	Information only.
DK1P02020S	P1061 RC not zero - terminating, RC= <return code #>	
DK1P02022S	TPM Request Mgr ATTACH failed - terminating, RC= <return code #>	
DK1P02024S	QEDIT1 macro failed - terminating, RC= <return code #>	
DK1P02026S	QEDIT2 macro failed - terminating, RC= <return code #>	
DK1P02027S	Product already running: Start terminating	Another copy of the tableBASE Process Manager is already running. It needs to be shut down before you can start it up again.

Error Code	Text	Meaning / Instructions
DK1P02028S	#FINDTPM macro failed - terminating, RC= <return code #>	
DK1P02029S	#FINDTPM macro failed - terminating, RC= <return code #>	
DK1P02030S	#FINDTPM macro defining TPVM compat failed, terminating, RC= <return code #>	
DK1P02032S	TPM CATALOG INVALID - TERMINATING	
DK1P02034S	Define for TPVM compat failed - terminating, RC= <return code #>	
DK1P02036S	#FINDTPM macro starting TPVM failed - terminating, RC= <return code #>	
DK1P02037S	Request Mgr failed - terminating	
DK1P02038S	TPM Request Mgr not in WAIT state - TPM terminating	
DK1P02039W	Unexpected POST to Request Mgr - terminating	
DK1P02040W	AUTOSTART for TPVM ,<tpvm name> failed	
DK1P02041I	TPVM <tpvm name> RE-LINKED TO TPM	The TPVM identified has been reconnected to the TPM.
DK1P02042S	#FINDTVC macro failed - terminating, RC = <return code #>TPVM = <tpvm name>	
DK1P02044I	TB <release name>: INITIALIZATION PROCESSING COMPLETED	Information only.
DK1P02046I	TPM detected Request Mgr completion- terminating	Information only.
DK1P02048I	TPM detected operator STOP command - terminating	Information only.
DK1P02050S	TPM detected bad POST - terminating, RM ECB = <control block #>, COMM ECB = <control block #>	
DK1P02052E	TPM unable to obtain service ENQ - AUTOSHUT not performed	
DK1P02054S	#FINDTPM macro failed - AUTOSHUT not performed, RC= <return code #>	
DK1P02056S	TPM Request Mgr not in WAIT state: AUTOSHUT cancelled	
DK1P02058E	AUTOSHUT for TPVM <tpvm name> failed	
DK1P02060S	#FINDTVC macro failed - terminating, RC = <return code #> TPVM = <tpvm name>	
DK1P02062S	#FINDTPM macro failed - TPM terminating; PVTE flags invalid, RC=<return code #>	
DK1P02063S	TPM "PM" ATTACH failed - RC= <return code #>	
DK1P02064S	VTS PROC can not be used to start a TPVM	
DK1P02065S	DETACH PM failed; RC= <return code #>	
DK1P02066E	DETACH macro failed - TPM termination continuing, RC= <return code #>	
DK1P02067S	License error - TPM terminating	
DK1P02068E	MODESET macro failed - TPM termination continuing, RC= <return code #>	
DK1P02069S	Corrupted navigation (#FINDTPM macro failed) - terminating	
DK1P02070I	TB <release name>: Processing completing	Information only.
DK1P02071S	TPM failed - unrecoverable error; terminating	
DK1P02072S	P1061: INITIAL ESTAEX ON FAILED - TERMINATING	
DK1P02074S	P1061: INPUT PARM ERROR - TERMINATING	
DK1P02076S	Program not authorized - terminating	Your load module is not in an authorized library.

Error Code	Text	Meaning / Instructions
DK1P02078S	MODESET macro failed - terminating	
DK1P02080S	SERVICE ENQ ERROR - TERMINATING	
DK1P02082W	WARNING: CURRENTLY ACTIVE DKL PRODUCTS: <product names>	
DK1P02084S	STORAGE macro OBTAIN failed - terminating, RC = <return code #>	
DK1P02086S	DIV macro IDENTIFY failed - terminating, RC =<return code #>	
DK1P02088S	DIV macro ACCESS failed - terminating, RC = <return code #>	
DK1P02090S	TPM LDS NOT INITIALIZED - TERMINATING	Your LDS for starting up the TPM has not been configured. See the *tableBASE Installation Guide* on how to configure it.
DK1P02092S	DSPSERV macro CREATE failed, RC = <return code #>	
DK1P02094S	ALESERV macro CREATE failed, RC = <return code #>	
DK1P02098S	DIV macro MAP failed, RC = <return code #>	
DK1P02100S	#FINDTPM macro failed - terminating, RC = <return code #>	
DK1P02101S	TPM pointer found with PRISTINE start - terminating	
DK1P02110S	IAZXJSAB macro failed - terminating, RC = <return code #>	
DK1P02112S	STORAGE macro OBTAIN failed - terminating, RC =<return code #>	
DK1P02114S	STORAGE macro OBTAIN failed - terminating, RC = <return code #>	
DK1P02115S	Invalid TPMDSECT address - terminating	
DK1P02116S	Failure setting recovery <state> IN <routine name> -TPM terminating, RC= <return code #>	<state> = ON or OFF
DK1P02117S	Corrupted Navigation (<target control block>)	
DK1P02118E	DIV macro, <macro name> failed - termination continuing, RC=<return code #>	
DK1P02120E	ALESERV macro DELETE failed - termination continuing, RC=<return code #>	
DK1P02121I	Recovery detected /C (cancel) command	Information only.
DK1P02122E	DSPSERV macro DELETE failed - termination continuing, RC=<return code #>	
DK1P02123S	Unexpected unrecoverable error	
DK1P02124E	STORAGE macro RELEASE for PVT failed - termination continuing, RC=<return code #>	
DK1P02125S	PC INIT failed - terminating, RC= <return code #>	
DK1P02126S	P1062: INITIAL ESTAEX ON FAILED - TERMINATING	
DK1P02128S	P1062: INITIAL FINDTPM FAILED - TERMINATING	
DK1P02130S	STORAGE macro OBTAIN failed for PC table - terminating, RC = <return code #>	
DK1P02131I	PREVIOUS LXRES VALUE FOUND - LX = <LX value>	Information only.
DK1P02132S	LXRES macro failed - terminating, RC = <return code #>	
DK1P02133I	LXRES VALUE BEING USED = <LX value>	Information only.
DK1P02134S	ETCRE macro failed - terminating, RC = <return code #>	

Error Code	Text	Meaning / Instructions
DK1P02136S	ETCON macro failed - terminating, RC = <return code #>	
DK1P02138S	LOAD macro failed - terminating, RC = <return code #>	
DK1P02140S	STORAGE macro OBTAIN for PC module failed - terminating, RC = <return code #>	
DK1P02142S	#FINDTPM macro failed - terminating, RC = <return code #>	
DK1P02144S	LOAD macro failed - terminating, RC = <return code #>	
DK1P02146S	STORAGE macro OBTAIN for PC module failed - terminating, RC = <return code #>	
DK1P02147S	#FINDTPM macro failed - terminating, RC = <return code #>	
DK1P02148S	Unexpected unrecoverable error	
DK1P02149I	Recovery detected /C (cancel) command	Information only.
DK1P02150	TPVM Start terminated or timed out; Review PROC, JCL or systems resources	The TPVM start could not be completed either due to errors in your user PROC or JCL or because there are insufficient system's resources to start another task. Review the TPVM job for JCL errors and check with your systems administrator for systems resource problems.
DK1P02151W	Initial recovery set on failed	
DK1P02152W	Failure setting recovery on, ROUTINE = <routine name #> RC = <return code #>	
DK1P02153W	Failure setting recovery off, ROUTINE = <routine name>, RC = <return code #>	
DK1P02154S	Initial #FINDTPM macro failed - terminating	
DK1P02155S	#FINDTPM macro failed, ROUTINE = <routine name>, RC = <return code #>	
DK1P02156S	TPM catalog not found or invalid	
DK1P02157E	Caller request code invalid, CODE = <code name>	
DK1P02158E	Request block data length incorrect	
DK1P02159E	Invalid TPVM name: all zeroes	The TPVM name cannot be all zeroes.
DK1P02160W	DEFINE requested for existing TPVM , <tpvm name>	
DK1P02161W	TPVM max entries exceeded in TPM catalog; DEFINE failed for TPVM <tpvm name>	There are a maximum of 14 entries allowed in the TPM catalog for named TPVMs and you have reached this limit.
DK1P02162S	DIV macro <macro name> failed, SERVICE = <service name>, RC = <return code #>, RS = <reason code #>	
DK1P02163W	START requested for running TPVM <tpvm name>	The TPVM that was requested to be started is already running.
DK1P02164E	START requested for undefined TPVM <tpvm name>	The TPVM that was requested to be started has not been defined.
DK1P02165E	START TPVM failed for TPVM <tpvm name>, RC=<return code #>	The TPVM start failed with the specified return code. Contact tableBASE technical support for further information.
DK1P02166E	Start TPVM ASCRE macro failed, TPVM =<tpvm name>, RC = <return code #>, RS = <reason code #>	
DK1P02167E	START TPVM terminated: TPVM=<tpvm name>, RC=<return code #>	
DK1P02168E	TPVM compat cannot be shut down using TPDRIVER SHUTDOWN command; use /P	The *compat* TPVM cannot be shutdown using TPDRIVER.

Error Code	Text	Meaning / Instructions
DK1P02169E	SHUTDOWN requested for TPVM not currently active, TPVM = <tpvm name>	The TPVM that was requested to be shutdown is not running.
DK1P02170E	SHUTDOWN request failed for TPVM <tpvm name>, TPVM not in WAIT state	
DK1P02171E	TPVM compat cannot be deleted	The *compat* TPVM definition cannot be deleted from the TPM catalog..
DK1P02172E	DELETE requested for undefined TPVM <tpvm name>	The TPVM that was requested to be deleted is not defined.
DK1P02173E	DELETE requested for running TPVM <tpvm name>	A TPVM that is running cannot be deleted.
DK1P02174E	TPVM <tpvm name> exceeds max TPVMs started in common memory	
DK1P02175I	TPM IS PROCESSING A REQUEST TO <action> TPVM <tpvm name>	Information only.
DK1P02176E	LDS in use; TPVM <tpvm name> not started	The LDS that is required to start the TPVM is already in use by another task.
DK1P02177I	Recovery detected /C (cancel) command	Information only.
DK1P02178I	LDS DSN=<LDS dataset name>	Information only.
DK1P02179S	Unexpected unrecoverable error	
DK1P02180S	Initial set recovery on failed - terminating	
DK1P02181S	P1070: INPUT PARM ERROR - TERMINATING	
DK1P02182S	Program not authorized - terminating	The program is not in an authorized library.
DK1P02183S	MODESET macro failed - terminating	
DK1P02184S	Corrupted navigation (<tpvm name>)	
DK1P02185S	TPM CONTROL BLOCK NOT FOUND	
DK1P02188I	Recovery detected /C (cancel) command	Information only.
DK1P02189S	TPM failed - unrecoverable error; terminating	
DK1P02190S	No parms received - terminating	
DK1P02200I	INITIALIZING TPVM <tpvm name>	Information only.
DK1P02201I	TPVM <tpvm name> IS RUNNING	Information only.
DK1P02202I	Operator STOP command detected for TPVM <tpvm name>	Information only.
DK1P02203I	TPVM <tpvm name> SHUTTING DOWN	Information only.
DK1P02204E	PROBLEM SAVING CATALOG FOR <tpvm name>	
DK1P02205E	PROBLEM UNMAPPING CATALOG FOR <tpvm name>, RC=<return code #>	
DK1P02206E	PROBLEM REMOVING CATALOG ACCESS FOR <tpvm name>, RC=<return code #>	
DK1P02207E	PROBLEM REMOVING CATALOG IDENTITY FOR <tpvm name>, RC=<return code #>	
DK1P02208E	Cannot delete ALET for <tpvm name>, RC=<return code #>	
DK1P02209E	Cannot delete dataspace for <tpvm name>, RC=<return code #>	
DK1P02210E	Cannot detach Request Manager for <tpvm name>, RC=<return code #>	
DK1P02211E	AUTOSTART for TPVM <tpvm name> ERROR STARTING VTS <vts name>	

Error Code	Text	Meaning / Instructions
DK1P02212E	AUTOSTART for TPVM <tpvm name>, VTS <vts name> FAILED, RC=<return code #>	
DK1P02213E	AUTOSHUT for TPVM <tpvm name>, VTS <vts name> FAILED, RC=<return code #>	
DK1P02214S	TPVM INIT failed - cannot open DDN DK1SNAP	
DK1P02215S	TPVM INIT failed - Program not authorized	
DK1P02216S	TPVM INIT failed - cannot change program mode	
DK1P02217S	TPVM INIT failed - TPM control block not found	
DK1P02218S	TPVM INIT failed - TPM not running	
DK1P02219S	TPVM INIT failed - error in QEDIT macro	
DK1P02220S	TPVM <tpvm name> INIT failed - cannot attach Request Manager	
DK1P02221S	TPVM <tpvm name> INIT failed - cannot determine shutdown type	
DK1P02222W	Failure setting recovery off	
DK1P02223S	TPVM failed - unrecoverable error	
DK1P02224I	TPVM <tpvm name> stopped running	Information only.
DK1P02225E	Error during shutdown - VTS not defined	
DK1P02226S	Messaging init request failed	
DK1P02227I	VTS <vts name> re-linked to TPVM <tpvm name>	Information only.
DK1P02231E	Problem saving catalog for <tpvm name>, RC=<return code #>	
DK1P02232S	DIV macro error, TPVM=<tpvm name>, RC=<return code #>	
DK1P02233S	Cannot allocate catalog dataspace for <tpvm name>, RC=<return code #>	
DK1P02234S	Cannot allocate ALET for <tpvm name> catalog, RC=<return code #>	
DK1P02235S	Cannot find TPM control block for <tpvm name>	
DK1P02236S	TPVM <tpvm name> exceeds max TPVMs started in common memory	
DK1P02237S	Cannot allocate space for <tpvm name> REQUEST BLOCK	
DK1P02238S	Cannot allocate space for <tpvm name> GCA	
DK1P02239S	TPVM <tpvm name> cannot use catalog for TPVM <tpvm name>	
DK1P02240E	Problem unmapping catalog for <tpvm name>, RC=<return code #>	
DK1P02241E	Problem removing catalog access for <tpvm name>, RC=<return code #>	
DK1P02242E	Problem removing catalog identity for <tpvm name>, RC=<return code #>	
DK1P02243E	Cannot delete dataspace for <tpvm name>, RC=<return code #>	
DK1P02244E	Cannot delete ALET for <tpvm name>, RC=<return code #>	
DK1P02245S	TPVM failed - unrecoverable error'	
DK1P02246S	Error loading PC for <tpvm name>, RC=<return code #>	

Error Code	Text	Meaning / Instructions
DK1P02247S	Error obtaining PC space for <tpvm name>, RC=<return code #>	
DK1P02248S	Error deleting PC for <tpvm name>, RC=<return code #>	
DK1P02249W	Failure setting recovery on, RC=<return code #>	
DK1P02250W	Failure setting recovery off, RC=<return code #>	
DK1P02251S	SHOWCB failed for LDS, RC = <return code #>	
DK1P02252S	TPVM LDS size allocation error; LDS size is: <LDS size>	
DK1P02253I	Recovery detected /C (cancel) command	Information only.
DK1P02261S	LDS for <tpvm name> was not empty as expected	
DK1P02262S	Definition not found for TPVM <tpvm name> in TPM catalog	The TPVM definition cannot be found in the TPM catalog.
DK1P02263S	Cannot find TPM control block for <tpvm name>	
DK1P02264S	TPVM <tpvm name> failed - unrecoverable error	
DK1P02265W	Failure setting recovery on, RC=<return code #>	
DK1P02266W	Failure setting recovery off, RC=<return code #>	
DK1P02267I	Recovery detected /C (cancel) command	
DK1P02301S	Cannot find VTS control block (GCA) for <tpvm name>	
DK1P02302S	Cannot find the catalog for <tpvm name>	
DK1P02303S	(1,C,8),' can not provide service requested opcode=<op code #>	
DK1P02304E	Invalid VTS definition under <tpvm name>, length=<length #>	
DK1P02305E	Invalid ALIAS definition under <tpvm name>, LENGTH=<length #>	
DK1P02306E	Maximum VTS definitions exceeded in <tpvm name> catalog	The maximum of 64 VTS definitions in the TPVM catalog has been reached,
DK1P02307E	Maximum ALIAS definitions exceeded in TPVM catalog <tpvm name>'	The maximum of 64 alias definitions in the TPVM catalog has been reached,
DK1P02308W	DEFINE requested for existing VTS <vts name> under <tpvm name>	The VTS definition already exists in the TPVM catalog.
DK1P02309E	DEFINE requested for existing <vts name> under <tpvm name>	The VTS definition already exists in the TPVM catalog.
DK1P02310E	An ALIAS cannot be used under TPVM compat	Information only.
DK1P02311E	No VTS associated with ALIAS <alias name> under <tpvm name>	
DK1P02312E	<vts name> IS RUNNING AS <alias name> under <tpvm name>	The VTS is running as the alias name specified.
DK1P02313E	Cannot find VTS definition <vts name> under <tpvm name>	The VTS definition cannot be found in the TPVM catalog.
DK1P02314E	Cannot find ALIAS definition <alias name> under <tpvm name>	The alias definition cannot be found in the TPVM catalog.
DK1P02315E	VTS <vts name> under <tpvm name> is not running	The VTS is not running under this TPVM.
DK1P02316E	ALIAS <alias name> under <tpvm name> is not running'	The alias name is not running under this TPVM.
DK1P02317E	VTS <vts name> under <tpvm name> is not Read-Only	The VTS specified is not in READ-ONLY mode.
DK1P02318W	VTS <vts name> under <tpvm name> is currently running	The VTS is already running under this TPVM.
DK1P02319E	ALIAS <alias name> under <tpvm name> is currently running	The alias name is already running under this TPVM.
DK1P02320S	VTS <vts name> under <tpvm name> is not in a wait state	

Error Code	Text	Meaning / Instructions
DK1P02321E	ALIAS <alias name> under <tpvm name> cannot be switched to ALIAS <alias name>	An alias name cannot be switched to another alias name.
DK1P02322E	VTS <vts name> under <tpvm name> is already associated with this ALIAS	The alias name is already associated with the VTS specified.
DK1P02323E	VTS <vts name> under <tpvm name> is associated with another ALIAS	The VTS specified is already associated with another alias name.
DK1P02324S	DIV macro error, TPVM=<tpvm name>, RC=<return code #>	
DK1P02325E	LDS in use, VTS <vts name> under TPVM <tpvm name> not started	The LDS that is required to start the TPVM is already in use by another task. See message 2326 for the name of the LDS.
DK1P02326E	LDS DSN=<LDS dataset name>	See message 2325 for the related message.
DK1P02327	VTS <vts name> under <tpvm name> was started by the VTS Agent	This VTS was started using VTSAgent.
DK1P02329I	Recovery detected /C (Cancel) command	Information only.
DK1P02330S	TPVM control block not found	
DK1P02331S	Service failed - unexpected unrecoverable error	
DK1P02332W	Failure setting recovery on, RC=<return code #>	
DK1P02333W	Failure setting recovery off, RC=<return code #>	
DK1P02335I	VTS <vts name> under <tpvm name> is now known as <alias name>	Information only.
DK1P02336I	VTS <vts name> under <tpvm name> no longer known as <alias name>	Information only.
DK1P02340I	<tpvm name> processing request to <action> <vts name>	Information only.
DK1P02341I	VTS Start terminated or timed out; Review PROC, JCL or systems resources	Problem with either PROC JCL or system resources: **PROC or JCL error**: look for a failed VTS job which would specify the reason for the failure. **Insufficient systems resources**: the start operation likely timed out, there would be no failed job to indicate this. Retry the start operation and if still unsuccessful, speak to your systems programmer regarding allocating more resources to your system.
DK1P02343I	P2063 TRMEXIT: Error in VTS Prolog	
DK1P02344I	P2063 TRMEXIT: PROC error - possible JCL error	
DK1P02345I	TPVM PROC cannot be used to start a VTS	
DK1P02400I	Initializing <xxxxxx> VTS <vts name> under <tpvm name>	Information only.
DK1P02401I	<xxxxxx> VTS <vts name> is running under <tpvm name>	Information only.
DK1P02402I	Operator cancelled VTS <vts name> under <tpvm name>	Information only.
DK1P02403S	Forced shutdown requested by recovery routine	
DK1P02404I	Shutdown request accepted for VTS <vts name> under <tpvm name>	Information only.
DK1P02405S	SNAP open failed	
DK1P02406S	MSG init failed	
DK1P02407S	TBLBASE init error	
DK1P02408S	TBLBASE term error	
DK1P02409S	Cancelled S222	
DK1P02410S	VTS start failed <vts name> - Program not authorized, RC=<return code #>	

Error Code	Text	Meaning / Instructions
DK1P02411S	VTS START failed <vts name> - Cannot change program mode, RC=<return code #>	
DK1P02412S	VTS START failed - TPVM name invalid name as supplied in hex <hex #>	
DK1P02413S	Cannot obtain ALET for <tpvm name>, RC=<return code #>	
DK1P02414S	Cannot find TPVM control block for <tpvm name>	
DK1P02415S	Cannot find catalog for <vts name> under <tpvm name>	
DK1P02416E	Cannot find VTS definition for <vts name>	
DK1P02417S	No LDS found for Read-Only VTS <vts name> under <tpvm name>	A READ-ONLY VTS must have an LDS associated with it.
DK1P02418S	LDSTSR DD statement not found or incorrect for <vts name> under <tpvm name>	The DD statement for the LDS is not found or incorrect. Check your VTS PROC.
DK1P02419S	VTS START failed under TPVM <tpvm name> - Error in QEDIT macro, RC=<return code #>	
DK1P02420S	Cannot find VTS control block for <vts name> under <tpvm name>	
DK1P02421S	Cannot load TBLBASE for VTS <vts name> under <tpvm name>	
DK1P02422S	RECOVERY ROUTINE DETECTED ERROR IN VTS PROLOG CODE	
DK1P02423S	VTS STARTUP FAILED - UNRECOVERABLE ERROR	
DK1P02424W	Failure setting recovery off, RC=<return code #>	
DK1P02425	ECB unknown	
DK1P02426	VTS <vts name> is running as ALIAS <alias name>	Information only.
DK1P02427	PVT Entry not found - VTS Startup terminated	
DK1P02500S	Messaging not available	
DK1P02590I	TPM Catalog configured with <count #> pages	Information only.
DK1P02591E	Configuration Module <module name> not loaded; RC=<return code #>	
DK1P02592E	Internal name <module name> does not match config module <module name>	
DK1P02593E	Version <version #> of config module <module name> does not match executing version <version #>	
DK1P02594E	<macro name> error for DDN TPMCAT, RC=<return code #>	
DK1P02595E	DISP for DDN TPMCAT must be OLD or NEW	
DK1P02596E	LDS DDN TPMCAT must be VSAM	The LDS for the TPM catalog must be a VSAM dataset.
DK1P02597E	Error in IBM macro <macro name>; RC=<return code #>, RS=<reason code #>	
DK1P02598E	LDS DDN TPMCAT too small at <# pages> pages; minimum is <# pages>	The LDS for the TPM catalog is too small.
DK1P02599S	TPM Catalog not configured due to preceding errors	
DK1P02601	<client name>	Information only.
DK1P02602	is not Licensed to use tableBASE Process Manager	You are not licensed to use tableBASE Process Manager.
DK1P02603	tableBASE Process Manager is Licensed to	Information only.
DK1P02604	Your "Process Manager" License expires on <date>	Information only.

Error Code	Text	Meaning / Instructions
DK1P02605	The PVT entry is not available	
DK1P02606	Your "Process Manager" License ends today	Information only.
DK1P02607	WARNING: Your "Process Manager" License will end in <number> days	Information only.
DK1P02608	Your "Process Manager" License expires in <number> days	Information only.
DK1P02609	NOTICE: Your "Process Manager" License has expired	Information only.
DK1P02610	is no longer Licensed to use tableBASE Process Manager	Information only.
DK1P02611	"Process Manager" must use an authorized LOAD library	Your library is not authorized.
DK1P02612	Licensing function found error in customer ID	
DK1P02613	Licensing function found error in product code	
DK1P02614	Licensing function found error; RC= <return code #>	
DK1P02615	Failure setting recovery on; RC= <return code #>	
DK1P02616	Failure setting recovery off; RC= <return code #>	
DK1P02617	Recovery detected /C (cancel) command	Information only.
DK1P02701S	TPVM <tpvm name> failed - Program not authorized	
DK1P02702S	TPVM <tpvm name> failed - cannot change program mode	
DK1P02703S	Cannot find control block for <tpvm name>	
DK1P02704S	Request Manager failed - unrecoverable error	
DK1P02705I	Recovery detected /C (cancel) command	Information only.
DK1P02706W	Failure setting recovery on, RC<return code #>	
DK1P02707W	Failure setting recovery off, RC=<return code #>	
DK1P02708S	Corrupted navigation path	
DK1P02709S	TPM control block not found	
DK1P02710S	TPM is not running	The tableBASE Process Manager PC Server is not running.
DK1P02711S	TPVM is not running	The TPVM is not running.
DK1P02800E	Error opening DDN TPDRCNTL, C error = <error code>	Error opening the input file for TPDRIVER.
DK1P02801E	Error reading input file, C error = <error code>	
DK1P02802E	Parameter value too long (<item name>)	The parameter value is too long.
DK1P02803E	Invalid command modifier (must be VTS or TPVM or ALIAS)	The valid TPDRIVER command modifiers are: VTS, TPVM or ALIAS.
DK1P02804E	Wrong input format - command incomplete and an EOF was reached	
DK1P02805E	Input command too long	The command line is too long.
DK1P02806E	Wildcard not allowed (<parameter name>)	Illegal wildcard use.
DK1P02807E	Invalid format for wildcard	The wildcard format is invalid.
DK1P02808E	Invalid format for default	
DK1P02809E	Invalid boolean value (<parameter name>)	Must be Y or N; nothing else is valid.
DK1P02810E	Invalid numeric value (<parameter name>)	Must be NUmeric value; i.e., 100K, 300000. Nothing else is valid.
DK1P02811E	Numeric value is out of bounds (<parameter name>)	Value must be within legal bounds. (For TSRSIZE, must be between 40K and 2G.)
DK1P02812E	Invalid value (RECOVERY)	

Error Code	Text	Meaning / Instructions
DK1P02813E	Missing parameter (<parameter name>)	The parameter name specified in parenthesis is required.
DK1P02814E	A parameter is invalid with this command (<parameter name>)	Command is not compatible with parameter used.
DK1P02815E	Wrong input format - cannot parse command	The input format is invalid.
DK1P02816E	Wrong input format	
DK1P02817E	Invalid command	The command is invalid.
DK1P02818E	Parameter is too long	The length of the parameter is too long.
DK1P02819E	Too many parameters in command, max is <count #>	Maximum is 20.
DK1P02820E	Invalid command modifier (must be VTS or TPVM or ALIAS)	The valid TPDRIVER command modifiers are: VTS, TPVM or ALIAS.
DK1P02821E	Command modifier invalid for this command	The command modifier specified for this command is invalid.
DK1P02822E	Invalid format for parameter <parameter name> (expected format: NAME1=Value1)	The format for the parameter should be as specified in paranthesis.
DK1P02823E	Invalid name for parameter <parameter name>	Parameter used is invalid.
DK1P02824E	Memory allocation failed	
DK1P02827W	TSRSIZE parameter will be ignored at VTS startup when an LDS is used	If an LDS is used for a VTS and the TSRSIZE parameter is also specified, it will be ignored and the actual TSRSIZE at startup will be that of the LDS.
DK1P02828W	TSRSIZE parameter will be ignored at VTS startup when TSRACCESS mode is Read-Only	If Read-Only mode is used for a VTS and the TSRSIZE parameter is also specified, it will be ignored and the actual TSRSIZE at startup will be that of the LDS.
DK1P02829E	LDS parameter required when TSRACCESS mode is Read-Only	If Read-Only mode is used for a VTS an LDS is required.
DK1P02830E	Invalid TSRACCESS value	The value specified for the TSRACCESS parameter is invalid. See *tableBASE Batch Utilities Guide* for details.
DK1P02831E	Invalid string format	
DK1P02832E	<command> command failed with error <error code> please review messages in <item> and <item> job	If there is no <item> job in which to search, see the specified error code in this table.
DK1P02900I	Process to remove and clean the product has started	Information only.
DK1P02901I	PVTE is full or pointer is corrupt	The PVT entry is full or the pointer to the PVTE has been corrupted. Contact tableBASE Technical Support for assistance.
DK1P02902I	Product vector table indicates product is not running	Information only
DK1P02903I	Product vector table indicates product is running	Information only.
DK1P02906I	Process to remove and clean has been run before	Information only
DK1P02907I	The product vector table pointer is corrupt	Contact tableBASE Technical Support for assistance.
DK1P02908W	Reply is incorrect. Halt is assumed	The response given to the WTO is incorrect, The process will be stopped.
DK1P02909I	Request to halt this process has been accepted	Information only.
DK1P02910I	Request to create a new PVT pointer has been accepted	Information only.
DK1P02911I	ENQ is not available. Process will terminate	
DK1P02912S	Messaging init request failed	
DK1P02913S	Program not authorized. Cannot proceed	
DK1P02914S	MODESET macro failed - terminating	

Error Code	Text	Meaning / Instructions
DK1P02915S	Error found between IBM control blocks CVT and CAT. Cannot proceed	Contact tableBASE Technical Support for assistance.
DK1P02916S	DKL slot in IBM CAT (Customer Anchor Table) is corrupt - zeroing entry	Contact tableBASE Technical Support for assistance.
DK1P02917I	Pointer to TPM control block is corrupt	Contact tableBASE Technical Support for assistance.
DK1P02918I	The TPM control block is corrupt	Contact tableBASE Technical Support for assistance.
DK1P02919I	Pointer indicates TPM control block does not exist	Contact tableBASE Technical Support for assistance.
DK1P02920I	TPVM control block is corrupt	Contact tableBASE Technical Support for assistance.
DK1P02921I	Process to remove and clean up <tpvm name> has started	Information only.
DK1P02922I	Forcing down TPVM <tpvm name>	Information only.
DK1P02923I	Pointer to the VTS control blocks is corrupt	Contact tableBASE Technical Support for assistance.
DK1P02924I	Shutting down and cleaning up <vts name> under <tpvm name>	Information only
DK1P02925I	Stopping PGM named: <program name> ASIDX=<asidx name>	Information only
DK1P02926I	PGM named: <program name> did not stop	
DK1P02927I	Corrupt or invalid PGM name - HEX value: <hex value #>	
DK1P02928I	Forcing down the TPM	Information only
DK1P02929I	Service ENQ <major name>, <minor name> in effect	Information only.
DK1P02930I	Process will halt based on the reply	Information only.
DK1P02931I	Process will continue as requested	Information only.
DK1P02932W	Failure setting recovery off, RC=<return code #>	
DK1P02933I	Recovery detected /C (Cancel) command	
DK1P02934S	Unexpected unrecoverable error	
DK1P02999I	FORCE ended	Information only.
DK1P03000E	Invalid command modifier (must be VTS or TPVM or ALIAS)	
DK1P03001E	Unaddressable Function	
DK1P03002E	Invalid command	No such command.
DK1P03003E	Invalid parameter list (TOKEN)	
DK1P03004E	Invalid token	
DK1P03006E	Invalid parameter list (OBJECT)	
DK1P03007E	Invalid command modifier (must be VTS or TPVM or ALIAS)	
DK1P03008E	Invalid parameter list (OBJECT NAME)	
DK1P03009E	Invalid value in one of the parameters	
DK1P03010E	Invalid parameter list (STRUCTURE)	
DK1P03011E	Structure is too large	
DK1P03012E	TBOPT/PARM34 length mismatch	
DK1P03013E	STACK POST routine failed	
DK1P03014E	TPM is not accessible	
DK1P03015E	TPVM is not accessible	
DK1P03016E	TPVM name does not match name in Request Block	
DK1P03017E	VTS name does not match name in Request Block	
DK1P03018E	Missing or invalid PROC name	PROC name is required for ths command and is either missing or the name does not exist in your PROCLIB.

Error Code	Text	Meaning / Instructions
DK1P03019E	Invalid AUTOSTART value used (must be Y or N)	Value for the AUTOSTART parameter must be Y or N.
DK1P03020E	Invalid TSRACCESS value used (must be RO or RW)	Value for the TSRACCESS parameter must be RO or RW
DK1P03021E	Invalid AUTOSHUT value used (must be Y OR N)	Value for the AUTOSHUT parameter must be Y OR N
DK1P03022E	TSRSIZE is less than minimum of 40K	Minimum value for TSRSIZE is 40K.
DK1P03023E	TSRSIZE is larger than maximum of 2G	Maximum value for TSRSIZE is 2G.
DK1P03024E	Request Block data is too short	
DK1P03025E	TPVM NOT FOUND OR NOT RUNNING	The TPVM cannot be found or it is not running.
DK1P03026	Product has no License; TPDRIVER terminated	You are not licensed to run tableBASE Process Manager
DK1P03027	The TPM control block is corrupt	Contact tableBASE Technical Support for assistance.
DK1P03040	Unaddressable object	
DK1P03041E	Invalid object	
DK1P03042I	Object name not found	
DK1P03043I	TPM data layout not valid	
DK1P03044E	TPVM data layout not valid	
DK1P03045E	VTS data layout not valid	
DK1P03046E	TPM catalog dataspace error	
DK1P03047E	TPVM catalog dataspace error	
DK1P03081E	P0100: PVT not found	
DK1P03082E	P0100: TPM not running	
DK1P03083E	P0100: ENQ error	
DK1P03084E	P0100: No parms passed	
DK1P03085E	P0100: TPM not found	
DK1P03086E	P0100: TPVM not found or not running	
DK1P03087E	P0100: TPVM not running	
DK1P03088E	P0100: Request Manager not waiting	
DK1P03089E	P0100: STACK POST routine failed	
DK1P03200E	DK1T03200E: no parms passed to message routine	
DK1P03201E	DK1T03201E: message INIT STORAGE OBTAIN error	
DK1P03202E	DK1T03202E: Message INIT OPEN error	
DK1P03203E	DK1T03203E: Target message not found	
DK1P03204E	DK1T03204E: Repository message length error	
DK1P03205E	DK1T03205E: Max nmbr of sub elements exceeded	
DK1P03206E	DK1T03206E: Max message length exceeded (text)	
DK1P03207E	DK1T03207E: Input data not half or full word	
DK1P03208E	DK1T03208E: Input substitution data exceeds max size	
DK1P03209E	DK1T03209E: Max msg length exceeded (SUBSTITUTION)	
DK1P03210E	DK1T03210E: Type X substitution data length error	
DK1P03211E	DK1T03211E: Message PUT failed	
DK1P03212E	DK1T03212E: Message FILE CLOSE error	
DK1P03213E	DK1T03213E: STORAGE RELEASE error	

Error Code	Text	Meaning / Instructions
DK1P03214E	DK1T03214E: Max WTO message length exceeded	

Note: Abnormal terminations between 99 and 1000 or over 1099, are tableBASE errors and should be brought to the immediate attention of your tableBASE Administrator.

tablesONLINE/CICS error messages

The following table contains information on the tablesONLINE/CICS error messages that can be encountered.

Table E-6: tablesONLINE / CICS messages and error codes

MSG #	Text	Meaning / Instructions
TB-2000 I	The program TBDRIVC terminated successfully.	Information only.
TB-2010 E	Transfer Error: Transfer transaction TRIN was not found.	TRIN is delivered as part of the TBOL product. Consult the tableBASE installation instructions.
TB-2011 E	Transfer Error: Invalid transaction.	Target transaction is not defined properly to CICS.
TB-2012 E	Transfer Error: User not authorized to execute the transaction.	Target transaction is not defined properly to CICS.
TB-2013 E	Transfer Error: Transaction is currently disabled.	Target transaction is not defined properly to CICS.
TB-2014 E	Transfer Error: TWA size too small.	Target transaction is not defined properly to CICS.
TB-2015 E	Transfer Error: Cannot retrieve PCT entry (Lock/Busy condition).	Target transaction is not defined properly to CICS.
TB-2016 E	Transfer Error: Target program not found or is currently disabled.	Target transaction is not defined properly to CICS.
TB-5000 E	Unexpected tableBASE error XXXX occurred while processing 'XX'.	Refer to the tableBASE error return codes for more information about errors processing command 'XX'. If problem not explained, contact DataKinetics tableBASE support.
TB-5001 I	Enter Down command to view additional rows.	Information only.
TB-5002 I	Enter Up command to view additional rows.	Information only.
TB-5003 W	Enter Help command for a display of # lines of messages.	Information only.
TB-5004 I	Enter Up or Down command to view additional rows.	Information only.
TB-5005 W	You only have the authority to browse. Edit capability disabled.	Information only.
TB-5006 W	You do not have the authorization to perform this function.	Information only. Contact tableBASE administrator.
TB-5007 E	Indirect help could not be done because table could not be opened.	See tablesONLINE/CICS User's Guide for further information.
TB-5008 E	Indirect help could not be done because key could not be found.	See tablesONLINE/CICS User's Guide for further information.
TB-5009 E	Command 'XXXXXXXXXXXXXXXXXXX' is not supported for hash organization.	See tablesONLINE/CICS User's Guide for further information.
TB-5010 E	Duplicate keys allowed accepts only a Y or N response.	
TB-5011 E	The generation number supplied is incorrect.	Occurs when the generation does not exist.
TB-5012 E	Library: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx not on XXXXXXXX table. Contact your system administrator.	Contact your tableBASE administrator.
TB-5013 E	Library selection failed. You are not authorized to modify 'xxxxxxxx'. Please contact the tableBASE system administrator.	Contact your tableBASE administrator.
TB-5014 E	Table open failed, you may not access tableBASE library: xxxxxxxx.	
TB-5015 E	Indirect help not followed. Indirect points to indirect help.	Only one level of indirect help is supported.

MSG #	Text	Meaning / Instructions
TB-5016 E	Invalid COMMAND: 'XXXXXXXXXXXXXXXXXXX' for multi-user update mode.	See tablesONLINE/CICS User's Guide for further information.
TB-5017 W	There are no other windows established, enter 'NEW' to create another Window.	Or you can enter NW on the command line.
TB-5018 W	This View restricts you to browse only. Edit capability disabled.	Contact your tableBASE administrator.
TB-5019 E	Table 'xxxxxxxx ' is accessed from a Read Only VTS region. Try Browse.	You accessed a table residing on VTS as a result of the VTS name appearing in the tableBASE library list. If you want to EDIT this table, your profile on the TBOLACT table must be changed. Contact your tableBASE administrator.
TB-5020 W	The last field has been searched. To repeat from start press FINDFLD.	The fields are searched from left to right or top to bottom starting from the cursor position. This message appears when the end of the fields have been reached.
TB-5021 W	The field name starting with ' ' is not found.	
TB-5023 E	View key field only accepts an 'Y' or 'N' entry.	
TB-5029 I	Changes to the view 'xxxxxxxx' have been cancelled.	Information only.
TB-5030 E	The count entered is outside the table. Count of last row is xxxxxxxx.	TBOL returns the last available row.
TB-5031 I	Changes to table 'xxxxxxxx' have been cancelled.	Information only.
TB-5032 I	The previous row examined has been deleted.	Information only.
TB-5033 I	The row has been added to the table.	Information only.
TB-5034 I	A row with this key is already on the table. The row has been added.	Information only.
TB-5035 I	The row has been updated.	Information only.
TB-5036 I	A row with this key is already on the table. The row has been updated.	Information only.
TB-5037 E	A row with this key is already on the table. Update rejected.	
TB-5038 W	The count entered is outside the table. You are positioned at bottom.	
TB-5039 E	The key entered does not match any row key on the table.	Re-enter command as required.
TB-5040 E	The exit program '********' could not be loaded.	Exits from TBOL are defined in one of two places: in the View of the table being edited/browsed or in the TBOLACT table if processing commands.
TB-5041 W	The count entered is outside the table. You are positioned at the top.	Re-enter command as required.
TB-5042 W	The first available row is:	Warning occurs when attempting to locate a row before the previously displayed rows and a user exit program suppresses the display of rows according to the exit logic.
TB-5043 W	The last available row is:	Warning occurs when attempting to locate a row after the previously displayed rows and a user exit program suppresses the display of rows according to the exit logic.
TB-5044 I	Table already in use. Multiple user access in effect.	Information only. Multi user access is a View attribute.
TB-5045 W	This row is currently in use by 'xxxxxxxx' in the 'aaaa' application.	When attempting to edit a row in use by another user xxxxxxxx.
TB-5046 I	Table accessed by multiple users has been closed.	Information only. Multi user access is a View attribute.

MSG #	Text	Meaning / Instructions
TB-5047 I	Multiple user access in effect. Others may be browsing or editing.	Information only. Multi user access is a View attribute.
TB-5049 E	Cannot close table 'xxxxxxxx'. The table is open for write.	Another TBOL Window , or an unrelated process is holding this table opened for write. Contact your system administrator.
TB-5050 E	The password supplied is invalid.	
TB-5051 W	Changes have been made to table 'XXXXXXXX'. To confirm that you indeed wish to cancel the changes: Select 'CANCEL' To avoid having all the changes lost: Select 'ENTER' or 'END'.	Information only. Follow message instructions as required.
TB-5052 W	Changes have been made to table 'XXXXXXXX'. To confirm that you indeed wish to save the changes and create a new generation: Select 'ENTER' or 'END' To cancel the changes that have been made: Select 'CANCEL'. To go back to editing without saving or canceling Enter 'RESHOW'.	Information only. Follow message instructions as required.
TB-5053 E	Changes have been made to table 'XXXXXXXX'. Changing the library, table, generation or password can only be done after work on the present generation is complete. In order to save the changes that have been made: Select 'ENTER' or 'END'. To cancel the changes that have been made: Select 'CANCEL'.	Information only. Follow message instructions as required.
TB-5054 W	A request to delete row(s) from the table has been entered. To confirm that you indeed wish to delete the rows: Select 'ENTER' or 'END'. To cancel the delete request: Select 'CANCEL'.	Information only. Follow message instructions as required.
TB-5055 I	Table 'XXXXXXXX' has been updated.	Information only.
TB-5056 I	View 'XXXXXXXX' has been updated.	Information only.
TB-5057W	You have made changes to table 'XXXXXXXX' which may have been updated by other users. Several users may be updating this table simultaneously. The 'CANCEL' command may not have any effect because another user may have saved the table. To undo any changes you made… In order to save your changes and any changes made by other users: Select 'ENTER' or 'END'.	Information only. Follow message instructions as required. Multi user access is a View attribute.
TB-5058 W	You have made changes to table 'XXXXXXXX' which may have been updated by other users. In order to save your changes and any changes made by other users: Select 'ENTER' or 'END' To avoid saving the changes and cycling another generation: Select 'CANCEL'. To go back to editing without saving: Enter 'RESHOW'.	Information only. Follow message instructions as required. Multi user access is a View attribute.

MSG #	Text	Meaning / Instructions
TB-5059 W	You have made changes to table 'XXXXXXXX' which may be being browsed by other users. Changing the library, table, generation or password can only be done after work on the present generation is complete. In order to save your changes and any changes made by other users: Select 'ENTER' or 'END'. To avoid saving the changes and cycling another generation: Select 'CANCEL'.	Information only. Follow message instructions as required. Multi user access is a View attribute.
TB-5060 W	You are not authorized to use the M2M command. Contact DataKinetics tableBASE support.	Contact DataKinetics tableBASE support.
TB-5061 W	A range of nnnnnn row(s) has been selected for deletion. Because some rows may be suppressed from view, the actual number to be deleted may be less. To confirm that you indeed wish to delete the rows: Select 'ENTER' or 'END'. To cancel the delete request: Select 'CANCEL'.	Information only. Follow message instructions as required.
TB-5062 E	Listing the contents of the VTS attached to xxxxxxxx is not supported.	Contact DataKinetics tableBASE support.
TB-5063 E	Internal error '9999' creating a directory listing.	Contact DataKinetics tableBASE support.
TB-5064 E	You are trying to create a duplicate primary key. Update rejected. Primary key fields are hi-lighted with a different color.	View Attribute
TB-5065 E	Delete enqueue error. Non-existing key: xxxxxxxxxxxxxxxxxxx	Record the Key. It may help pinpoint an issue.
TB-5066 E	The row with primary key: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx has been deleted or moved by another user.	Record the Key. It may help pinpoint an issue.
TB-5500 W	Exit program called with invalid indicators:	User Exit error. Contact tableBASE administrator.
TB-5501 W	Field name is spaces.	
TB-5502 W	To indicate a key field code Y. Otherwise leave blank or code N. If the field is a Dynamic View Suffix, code S. If it is also a key field as well as the Dynamic View Suffix, code B. If this is a View for Editing a table using an Alternate Index and the Primary Key of the table is in a different location, code P. If a Primary Key Field and an Alternate Index Key Field overlap, code X.	
TB-5503 E	Invalid display or internal table format.	With Cursor on field, press Help (PF1).
TB-5504 W	Display length must be xxxx and has been substituted.	
TB-5505 E	Display length must be in the range xxxx through xxxx.	With Cursor on field, press Help (PF1).
TB-5506 W	Internal table length must be xxxx and has been substituted.	
TB-5507 E	Internal table length must be in the range xxxx through xxxxx.	With Cursor on field, press Help (PF1).
TB-5508 E	Invalid edit length code 'X' on TBOLHKFM table. Valid values are: blank, D, E, H, N, P, Q, 0.	
TB-5509 E	Invalid attribute. Pick N, C, F, D, Q, S, P, p, M, m, V, v, or blank.	With Cursor on field, press Help (PF1).
TB-5510 E	Invalid feature, must be D, N, Y, ':' or blank.	With Cursor on field, press Help (PF1).
TB-5511 E	Invalid action code. It must be Y, N, R, E, I, D, C, U, or blank.	With Cursor on field, press Help (PF1).
TB-5512 W	Program name 'xxxxxxxx' does not exist.	
TB-5513 E	Invalid direction, must be I for inbound to display or O for outbound.	With Cursor on field, press Help (PF1).
TB-5514 E	Invalid timing, must be B for before system action or A for after.	With Cursor on field, press Help (PF1).

MSG #	Text	Meaning / Instructions
TB-5515 E	Key fields must be contiguous. Check field ====>	
TB-5516 E	Display length must equal internal table length.	With Cursor on field, press Help (PF1).
TB-5517 E	Display length should equal table length or table length + 1.	With Cursor on field, press Help (PF1).
TB-5518 E	Display length should equal (2 * table length). Subtract 1 if no sign.	With Cursor on field, press Help (PF1).
TB-5519 E	Display length must equal 2 * table length.	With Cursor on field, press Help (PF1).
TB-5520 E	Display length should equal table length + 1 or table length + 2.	With Cursor on field, press Help (PF1).
TB-5521 E	Display length should equal (2 * table length). Add 1 if signed.	With Cursor on field, press Help (PF1).
TB-5522 W	Exit program entered without setting exit indicators.	User Exit error. Contact tableBASE administrator.
TB-5523 E	You may not move a field to a position before a key field.	In display mode of View edit the keys are always grouped first .
TB-5524 E	You cannot add new physical fields in display mode.	
TB-5525 E	You may not move a key field.	
TB-5526 E	You cannot delete a field in display mode.	
TB-5527 E	The number of key fields must be between 1 and 50 inclusive.	
TB-5528 E	The total length of the key fields must not be greater than 256.	
TB-5529 W	Application specific help table 'XXXXXXXX' not found on any library.	
TB-5530 E	Invalid field location for field: View may be corrupt - contact your technical support person.	If the error cannot be explained, contact DataKinetics tableBASE support.
TB-5531 E	The Data Table is currently in use.	Not used in Version 6 TBOL
TB-5532 W	Data/Index 'xxxxxxxx' requires RSZ, KSZ, KLOC: nnnnnnnnnnnnnnnnnn, in order to conform to the View just defined. Change the values of the row size, key size & location of the physical Data Table to conform to the View. If this is a View for an Alternate Index the warning above may not apply. Please note the KEY SIZE, and KEY LOCATION so that these values can be used to verify the Alternate Index Definition - Option 6 on the Define Table and View menu.	
TB-5533 E	Data Table 'xxxxxxxx' is in use - try again later.	Occurs when attempting to edit a table.
TB-5534 E	You cannot change the keys or suffix in display mode.	For some applications, a different display order is possible. Contact DataKinetics tableBASE support.
TB-5535 E	You cannot change fields to be key fields while in display mode.	Go to DEFINE VIEW on the "Define Table and View" menu to change key fields.
TB-5536 E	The total length of the key fields must be greater than zero.	
TB-5537 E	Number of delimiters must be equal the display length.	
TB-5538 E	The display mask must include a valid sign symbol.	This applies to numeric Display Formats N and 0 through 9.
TB-5539 E	The display mask must have 'X' decimal places.	This has to match the number of decimals places of the Display Format
TB-5540 E	The number of delimiters must be one less than the display length.	
TB-5541 E	The trigger field name supplied is not valid. A trigger field must be defined with an action code of Y, R or E.	Verify the spelling of the Trigger field name. It must be a field already defined in the View.
TB-5542 E	Invalid decimal point encountered at position 'XX'.	
TB-5543 W	Edit pattern only allowed for display format of 'X'.	
TB-5544 W	Invalid or incomplete pattern delimiter encountered at position 'XX'.	

MSG #	Text	Meaning / Instructions
TB-5545 E	Number of edit pattern symbols must be equal to display length.	
TB-5546 E	View 'XXXXXXXX' is in use by another application.	Someone else is editing the View.
TB-5547 E	The field format and/or length differs from source key field.	
TB-5548 E	Source field format and/or length differs from importation field.	
TB-5549 E	Source field name required for duplication action.	
TB-5550 E	The source field name entered already has a duplication action defined	If you want multiple duplicates, always take the value from a single source field.
TB-5551 E	Source field format and/or length differs from duplication field.	
TB-5552 E	Create date or update date action codes apply only to date fields.	
TB-5553 E	The source field is not a key field as required for action Y, R or E.	
TB-5554 E	The source field specified is not the entire key of the corresponding source Data Table.	
TB-5555 E	Source field name not defined in the view.	
TB-5556 E	Importation action requires a previously defined field with an action code of Y, R or E.	
TB-5557 E	Internal view alternate table access failed.	Contact DataKinetics tableBASE support.
TB-5558 E	Source field must be the entire key of the Source Table.	
TB-5559 E	A FIELD FORMAT of P, F, or H cannot be used as a dynamic View suffix.	Only formats that define displayable characters are supported.
TB-5560 E	The source field name entered must not be the same as the target field	
TB-5561 A	A row on view 'xxxxxxxx' cannot be found. Internal error. Abort task.	Contact DataKinetics tableBASE support.
TB-5562 E	tableBASE error 'XXXX' occurred creating internal table 'RSXXXXXX'.	You may have too small a TSR (tableSPACE Region).
TB-5563 E	tableBASE error 'XXXX' occurred storing internal table 'RSXXXXXX'.	Target library is Read Only.
TB-5564 E	The total length of all fields: 99999 exceeds 32,763 bytes.	32,763 bytes is the maximum row size.
TB-5565 W	The attribute V or v cannot be used with display format Y. The attribute has been set to a space.	
TB-5566 E	The target field must be 8 bytes to receive a DDNAME from the list or it must be 44 bytes to receive a DATASET NAME.	
TB-5567 E	The target field must be 8 bytes to receive a table or view name.	
TB-5568 E	More than one dynamic suffix. Check field ====>	
TB-5569 E	The total length of the Primary key fields must be less than 257.	The sum of the lengths of all fields comprising a key must be 256 or less.
TB-5570 E	The FIELD LENGTH/FORMAT must be '8' 'X' or 'U' respectively for the insert USERID action code 'B'	
TB-5571 E	'xxxxxxxx' is not a valid TBOL view.	Verify view, and re-issue.
TB-5600 W	Exit called for a table it doesn't know about. Table is xxxxxxxx.	
TB-5601 E	You cannot delete your own session. Action ignored.	Information only.
TB-5700 W	Utility processing is bypassed as the current operating mode is 'X'.	Contact DataKinetics tableBASE support. Very likely the XXXXMENU table has errors.
TB-5701 W	Invalid action. Use S to select a utility.	Information only. Follow message instructions as required.
TB-5702 W	Table 'xxxxxxxx' has been copied as 'xxxxxxxx'.	Information only.
TB-5703 W	Table 'xxxxxxxx' has been renamed to 'xxxxxxxx'.	Information only.

MSG #	Text	Meaning / Instructions
TB-5704 W	Table 'xxxxxxxx' has been deleted.	Information only.
TB-5705 W	View 'xxxxxxxx' has been copied as 'xxxxxxxx'.	Information only.
TB-5706 W	View 'xxxxxxxx' has been renamed to 'xxxxxxxx'.	Information only.
TB-5707 W	View 'xxxxxxxx' has been deleted.	Information only.
TB-5708 W	A generation of table 'xxxxxxxx' has been deleted.	Information only.
TB-5709 W	User profile has been updated.	Information only.
TB-5710 W	Application table 'xxxxxxxx' has been copied as 'xxxxxxxx'.	Information only.
TB-5711 E	Function cannot be performed: Table 'xxxxxxxx' not found.	
TB-5712 E	Utility failed: Table 'xxxxxxxx' is currently in use.	Try again later, or contact tableBASE administrator.
TB-5713 E	Utility failed: Table 'xxxxxxxx' already exists on library.	
TB-5714 E	Utility: View 'xxxxxxxx' parameters not currently defined.	
TB-5715 E	Utility failed: view 'xxxxxxxx' already exists on the library.	
TB-5716 E	Utility cannot be performed: View 'xxxxxxxx' not found.	
TB-5717 E	Utility failed: Prefix required to generate a new set of tables.	
TB-5718 E	Utility failed: Invalid tableBASE library. DDNAME: xxxxxxxx.	
TB-5719 W	DDNAME xxxxxxxx not on 'xxxxxxxx' - contact your system administrator.	Contact your tableBASE administrator.
TB-5720 E	Utility failed: Invalid password supplied.	Try again, or contact tableBASE administrator.
TB-5721 W	Table 'xxxxxxxx' has been defined.	Information only.
TB-5722 W	Table 'xxxxxxxx' definition has been updated.	Information only.
TB-5723 W	Supplemental information of view 'XXXXXXXX' has been updated.	Information only.
TB-5724 E	Reducing row size may result in data loss. Press execute to continue.	Information only. Follow message instructions as required.
TB-5725 E	The passwords for table 'xxxxxxxx' have been changed.	Information only.
TB-5726 W	Alternate Index 'xxxxxxxx' has been defined for table 'xxxxxxxx'.	Information only.
TB-5727 E	Table 'xxxxxxxx' from a different library is currently in use.	
TB-5728 E	Illegal To prefix 'xxxx'.	
TB-5729 E	Blank table name. This is a required entry.	
TB-5730 E	Invalid generation.	
TB-5731 E	Function not performed - tableBASE library XXXXXXXX is full.	
TB-5732 E	tableBASE library must be entered.	
TB-5733 E	The RSZ, KSZ & KLOC have been updated to conform to an updated view. If these values are not required and the original values of the data are to be used, select ENTER again immediately upon return. Press EXECUTE to change the Data Table to conform to the VIEW. This could result in data loss if you are reducing the row size.	Information only. Follow message instructions as required.
TB-5734 W	Table is not an alternate. Fields filled in from the Data Table.	
TB-5735 E	Table 'xxxxxxxx' is an alternate for Data Table 'xxxxxxxx'.	
TB-5736 W	The RSZ, KSZ & KLOC have been filled in from the view.	
TB-5737 W	View 'xxxxxxxx' could not be found and was not renamed.	
TB-5738 W	Alternate Index 'xxxxxxxx' has been updated for table 'xxxxxxxx'.	

MSG #	Text	Meaning / Instructions
TB-5739 E	New name cannot be the same as old name.	
TB-5740 E	This operation is not supported for tables from/to the VTS server.	
TB-5741 E	Alternates should not be updated with this option. Use option 6.	The name used was that of an Alternate Index.
TB-5742 W	Utilities function error. There is no processing logic for row ID 'X'.	Contact DataKinetics tableBASE support.
TB-5743 E	Utility failed: Invalid password supplied for target.	
TB-5744 E	The SYSIN parameter starting with ' xxxxxxxx . . . . ' cannot be found.	There is an error in the TBOLJCLx template table. Cannot identify the JCL statement. Contact DataKinetics tableBASE support.
TB-5745 E	The JES spool data set could not be opened, reason code '9999'.	Contact your CICS system administrator.
TB-5746 E	The JES spool data set could not be closed, reason code '9999'.	Contact your CICS system administrator.
TB-5747 E	The JES spool data set could not be written to, reason code '9999'.	Contact your CICS system administrator.
TB-5748 E	The JCL template table, 'xxxxxxxx' does not exist. Please re-enter.	Re-enter command as required.
TB-5749 E	The variable ' ' in JCL statement 00000000 was not replaced.	There is an error in the TBOLJCLx template table. The template JCL statement is missing a search for value. Contact DataKinetics tableBASE support.
TB-5750 E	The JCL statement starting with '//xxxxxxxx . . . . ' cannot be found.	There is an error in the TBOLJCLx template table.
TB-5751 I	Job 'xxxxxxxx' has been submitted to generate copybook 'mmmmmmmm'.	Information only.
TB-5752 E	The generated JCL from template 'xxxxxxxx' for table/view 'tvtvtvtv'has been submitted with an error. Please examine the JES queue to ensure that the generated JCL contain the correct program name and that the keyword substitutions have been accurately performed for this function.	Examine the JES queue and determine the error before re-submission. You may need to change the accounting information.
TB-5753 I	Job 'xxxxxxxx' has been submitted to print table 'tttttttt'.	Information only.
TB-5754 E	Alternate definition failed: Table 'xxxxxxxx' already exists. Use Execute with 'UPDATE' on command line to replace alternate table.	Information only. Follow message instructions as required.
TB-5755 E	Copy cannot be performed: Table 'xxxxxxxx' exists on target library.	
TB-5756 E	Update of alternate failed: Alternate table 'xxxxxxxx' does not exist.	
TB-5757 E	Copy cannot be performed: View 'xxxxxxxx' exists on target library.	
TB-5758 I	Job 'xxxxxxxx' has been submitted to print View definition 'vvvvvvvv'.	Information only.
TB-5759 I	The alternate definition name 'xxxxxxxx' is unavailable - (not found).	Information only.
TB-5760 E	Enter the name of a table that contains the restructuring rules.	Follow message instructions as required. This table would have been saved from a prior visit to the Data Restructuring Utility.
TB-5761 E	Re-enter a different name.	Re-enter a different name as required.
TB-5762 E	The restructuring table, 'xxxxxxxx' does not exist. Please re-enter.	Re-enter restructuring table name as required.
TB-5763 E	Enter Y or N to indicate the generations you want to restructure.	Information only. Follow message instructions as required.

MSG #	Text	Meaning / Instructions
TB-5764 I	The restructuring table, XXXXXXXX is not a valid restructuring table.	Information only. The restructuring table contains information to restructure tables. It can only be processed by TBOL.
TB-5765 E	Restructuring tbl: XXXXXXXX requires RSZ/KSZ/ KLOC:nnnnnnnnnnnnnnnnnn. The Data Table that has been entered cannot be restructured with this restructuring rules table.	The Restructuring table does not match the basic information of the data table that you are attempting to restructure.
TB-5766 E	Generation (-x) of the data has RSZ/KSZ/KLOC: nnnnnnnnnnnnnnnnnn. This does not conform to the latest generation. Restructuring aborted. Restructuring is possible with the most recent generation. Set the All Generations flag to N so only the most recent is restructured.	Older generations of the data table do not have the same layout as the latest generation from which the restructuring data was taken from.
TB-5767 I	N generation(s) of the table 'XXXXXXXX' has been restructured.	Information only.
TB-5768 I	The restructuring rules are saved in the table 'xxxxxxxx'.	Information only.
TB-5769 E	If the restructuring rules are to be saved for future use with other tables, enter a restructuring table name.	Information only. Follow message instructions as required. The name should be recorded so it can be used at a later time or deleted if it is never used again.
TB-5770 I	The data is already restructured. Use END or CANCEL to exit utility.	Information only.
TB-5771 E	Although you have elected not to restructure the Data Table(s) at this time, it is essential to enter a name to save the restructuring table for use at a later date. If this is not a requirement, enter CANCEL to avoid a repeat of this message.	Information only. Follow message instructions as required.
TB-5772 E	FLD: xxxxxxxxxxxxxxxxxxxx Error: One of the rows from the existing table cannot be converted.	
TB-5773 E	Data Table 'XXXXXXXX' cannot be stored, library 'XXXXXXXX' is full. Free up some space on the target library, return to this option (7), and try again.	Open up another Window and delete an unused table to free up some space
TB-5774	The name 'XXXXXXXX' entered is either an alternative index or a paged table and cannot be restructured directly. For alternate indexes restructure the underlying base Data Table. For a paged table, change it to an in-memory table using the utility TBEXEC.	For alternate indexes: Restructure the underlying base Data Table. For a paged table: Change it to an in-memory table using the utility TBEXEC.
TB-5775 E	The variable ' ' in SYSIN statement 00000000 was not replaced.	There is an error in the TBOLJCLx template table. The template JCL statement is unable to replace a search for value. Contact DataKinetics tableBASE support.
TB-5777 W	The Multi-User table update option requires unique primary keys. If this is the VIEW for a table (not an Alternate Index), set the DUPLICATE KEY IND to 'N'. Press HELP for more details.	Follow message instructions as required.

MSG #	Text	Meaning / Instructions
TB-5778 E	The Multi-User table update option requires unique primary keys.	With Multi-User updates, TBOL cannot process tables that have duplicate primary keys.
	If this is the VIEW for an Alternate Index, set the DUPLICATE PRIME KEYS to 'N' when MULTIPLE USER UPDATE is set to 'M'.	With Multi-User updates, TBOL cannot process tables that have duplicate primary keys.
TB-5801 E	## - Action Error:	The ## identifies the field that has the Error: xxx.. Often pressing help gives you the list of values to pick from.
TB-5802 E	## - Import Error:	The ## identifies the field that has the Error: xxx..
TB-5803 E	## - Duplication Error:	The ## identifies the field that has the Error: xxx..
TB-5804 E	## - Mandatory Error: An EOF erase does not constitute a valid key stroke. Spaces are valid.	The ## identifies the field that has the Error: xxx..
TB-5805 E	## - Pattern: 'XXXXXXXX' does accept data as entered. Refer to legend below: Z - alphabetic; A - alphabetic or blank; 9 - numeric; I - numeric or blank; Y - alphabetic or numeric; X - alphabetic, numeric or blank; B - blank; C - any character (no validation); Literals - any set of characters between a pair of (!) marks. Use Help (F1) in Define View Option for more details.	The ## identifies the field that has the Error: xxx..
TB-5806 E	## - Verification Error:	The ## identifies the field that has the Error: xxx..
TB-5811 E	The source View 'xxxxxxxx' not found.	The Source View is required to locate the import data.
TB-5812 E	The source Data table 'xxxxxxxx' not found.	Data importation is not possible.
TB-5813 E	The source table 'xxxxxxxx' is password protected.	Data importation is not possible.
TB-5814 E	tableBASE error 'xxxxx' occurred opening Source Table 'xxxxxxxx'.	Refer to the tableBASE error return codes for more information.
TB-5815 E	Source Table, 'xxxxxxxx' must be in descending order for x' action.	Data importation is not possible.
TB-5816 E	The length of field 'xxxxxxxxxxxxxxxx' and Source Table 'xxxxxxxx' key field length do not match.	Data importation is not possible.
TB-5817 E	Source field 'xxxxxxxxxxxxxxxx' not found in table 'xxxxxxxx'.	Data importation is not possible.
TB-5818 E	Source field 'xxxxxxxxxxxxxxxx' in table 'xxxxxxxx' is not a key.	Data importation is not possible.
TB-5819 E	Source field 'xxxxxxxxxxxxxxxx' in source view 'xxxxxxxx' has a different length than the field defined in the view.	Data importation is not possible.
TB-5820 E	Source field 'xxxxxxxxxxxxxxxx' in source view 'xxxxxxxx' is not the entire key of the corresponding source Data Table.	Data importation is not possible.
TB-5821 E	Source field 'xxxxxxxxxxxxxxxx' in source view 'xxxxxxxx' has a different format than the field defined in the view.	Data importation is not possible.
TB-5822 E	tableBASE error 'xxxx' occurred creating internal table 'SPxxxxx'.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5823 E	tableBASE error 'xxx' occurred emptying internal table 'ECxxxxxx'.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5824 E	tableBASE error 'xxxx' occurred creating internal table 'ECxxxxxx'.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.

MSG #	Text	Meaning / Instructions
TB-5825 E	tableBASE error 'xxxx' occurred deleting internal table 'ECxxxxx'.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5826 E	tableBASE error 'xxxx' occurred deleting internal table 'SPxxxxx'.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5827 E	Upper range 'xxxxxxxxxxxxxxxx' in source view 'xxxxxxxx' has a different length than the lower range field.	Data importation is not possible.
TB-5828 E	Upper range 'xxxxxxxxxxxxxxxx' in source xxxxxxxx' has a different format than the lower range field.	Data importation is not possible. Fix the format mismatch first.
TB-5829 E	Lower range 'xxxxxxxxxxxxxxxx' in source view 'xxxxxxxx' has no corresponding upper range field for action R.	Data importation is not possible. Fix the
TB-5830 E	Internal get storage failed for data importation processing.	Data importation is not possible.
TB-5832 E	View data importation pointers corrupted.	Contact DataKinetics tableBASE support.
TB-5833 E	View data duplication pointers corrupted.	Contact DataKinetics tableBASE support.
TB-5834 E	Upper range in source Data Table 'xxxxxxxx' is too short or missing.	Data importation is not possible.
TB-5835 E	Initial value does not conform to the display format or edit pattern.	
TB-5836 E	Highest bound does not conform to the display format or edit pattern.	
TB-5837 E	Lowest bound does not conform to the display format or edit pattern.	
TB-5838 E	The Lowest bound value is greater than Highest bound value.	
TB-5839 E	Both the Highest and Lowest bound values must be specified.	
TB-5840 E	The Initial value is not between the Highest and Lowest bounds.	
TB-5841 E	Initial value has longer length than the display length.	
TB-5842 E	Lowest bound value has longer length than the display length.	
TB-5843 E	Highest bound value has longer length than the display length.	
TB-5844 E	## - Value Lower than bound: 'lllllllllllllllllllllllllllllllll'. Please enter a value greater than the lower bound indicated.	The ## identifies the field that has the Error. Enter a value greater than the lower bound indicated.
TB-5845 E	## - Value Greater than bound: 'uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu'. Please enter a value lower than the upper bound indicated.	The ## identifies the field that has the Error. Enter a value lower than the upper bound indicated.
TB-5897 E	QIDERR error: command code 'xxxx' response code 'xxxxxxxxxxx'. A CICS related operational resource is unavailable. The table directory listing was aborted. Please try HELP again (PF1).	The table directory listing was aborted. Please try HELP again (PF1).
TB-5898 E	QIDERR error: command code 'xxxx' response code 'xxxxxxxxxxx'. A CICS related operational resource is unavailable. You cannot continue to edit the table. Table rows may have been updated. In order not to lose any changes, You must exit the table editor and save any changes. After exiting this message screen, press END (PF3) from the Identify Table/Row screen.	Exit the table editor and save any changes. After exiting this message screen, press END (PF3) from the Identify Table/Row screen.
TB-5899 E	QIDERR error: command code 'xxxx' response code 'xxxxxxxxxxx'. A CICS related operational resource is unavailable. You cannot continue to edit/browse the table. After exiting this message screen, press END (PF3) from the Identify Table/Row screen and try editing or browsing the table at a later time.	Try editing or browsing the table at a later time.

MSG #	Text	Meaning / Instructions
TB-5900 W	Help message unavailable, message 'xxxxxxx' not on table 'xxxxxxxx'.	
TB-5901 E	Unable to open table 'xxxxxxxx', reason code 9999.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5902 E	Unable to invoke alternate table 'xxxxxxxx', reason code 9999.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5903 W	Help information for table 'xxxxxxxx' is not found on table 'xxxxxxxx'	
TB-5904 W	DESC: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx' not on 'xxxxxxxx'	This could occur when a DESCription with key has been deleted.
TB-5905 A	Error creating table 'XXXXXXXX', reason code 9999. Job aborted.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5906 W	Tutorial data 'xxxxxxxxxxxxxxxxxxxxxxx' not found on table 'xxxxxxxx'.	
TB-5907 E	This PF key has not been defined on the PFK table.	
TB-5908 W	This application does not have items defined on menu table 'xxxxxxxx'.	Refer to the tablesONLINE/CICS User's Guide for creation of Menu Items.
TB-5909 W	Help data 'xxxxxxxxxxxxxxxxxxxxxxx' not on table 'xxxxxxxx'	Refer to the tablesONLINE/CICS User Guide for creation of Menu Items.
TB-5910 E	Transfer program 'xxxxxxxx' of menu option 'xxxx' not found/ disabled.	Refer to the tablesONLINE/CICS User Guide for creation of Menu Items.
TB-5911 E	Blank transfer parameters for menu option 'XXXX'	Refer to the tablesONLINE/CICS User Guide for creation of Menu Items.
TB-5912 E	Invalid COMMAND or OPTION: 'xxxxxxxxxxxxxxxxxxxxxxx', please re-enter.	Re-enter command as required.
TB-5913 E	Transfer option 'xxxxxxxxxxxxxxx' is not found on menu table 'xxxxxxxx'.	Refer to the tablesONLINE/CICS User Guide for creation of Menu Items.
TB-5914 W	The value entered for this command should be numeric.	Re-enter command as required.
TB-5915 W	Additional tutorial information is not found on table 'xxxxxxxx'.	
TB-5916 E	The keys are protected. The row cannot be deleted in this mode.	
TB-5917 E	Command or option 'xxxxxxxxxxxxxxx' not supported for browse.	See tablesONLINE/CICS User's Guide for further information.
TB-5918 E	Command or option 'xxxxxxxxxxxxxxx' unsupported for hash tables.	See tablesONLINE/CICS User's Guide for further information.
TB-5919 E	There are no rows on the table.	
TB-5920 E	View 'xxxxxxxx' cannot be found on the library searched.	
TB-5921 E	View 'xxxxxxxx' is empty.	
TB-5922 E	Unable to open view 'xxxxxxxx', reason code 9999.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5923 W	Dynamic open of view 'xxxxxxxx' failed, reason code 9999.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5924 W	Dynamic open failed. View 'xxxxxxxx' not found.	
TB-5925 E	tablesONLINE system table 'xxxxxxxx' is not on any library searched.	

MSG #	Text	Meaning / Instructions
TB-5926 E	Unable to open tablesONLINE system table 'xxxxxxxx', reason code 9999.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5927 E	tablesONLINE system table 'xxxxxxxx' is empty.	
TB-5928 E	tablesONLINE system table 'xxxxxxxx' is in use by another application.	
TB-5929 W	Dynamic open failed. View 'xxxxxxxx' is empty.	
TB-5930 E	## - Conversion Error:	The ## identifies the field that has the Error: xxx..
TB-5931 E	## - Conversion Error:	The ## identifies the field that has the Error: xxx..
TB-5932 E	Key field ## - Conversion Error:	The ## identifies the field that has the Error: xxx..
TB-5933 E	View 'xxxxxxxx' is in use by another application.	
TB-5934 E	Data Table 'xxxxxxxx' cannot be stored, library 'xxxxxxxx' is full. In order not to lose the work just completed, enter another Window and free up some space by deleting unnecessary tables. Once this is done: Select 'ENTER' or 'END'. To cancel the changes that have been made: Select 'CANCEL'.	Follow message instructions as required.
TB-5935 E	Data Table 'xxxxxxxx' is in use by another application. Try browse.	
TB-5936 E	View 'xxxxxxxx' cannot be stored, library 'xxxxxxxx' is full. In order not to loose the work just completed, enter another Window and free up some space by deleting unnecessary tables. Once this is done: Select 'ENTER' or 'END'. To cancel the changes that have been made: Select 'CANCEL'.	Follow message instructions as required.
TB-5937 E	View defines row size of 00000 which differs from table RSZ of 00000.	In order to use this View, you must make the Key size, Row size and Key location match the Data table.
TB-5938 E	View defines key location 00000 which differs from tbl location 00000.	In order to use this View, you must make the Key size, Row size and Key location match the Data table.
TB-5939 E	View defines key size of 0000 which differs from table key of 0000.	In order to use this View, you must make the Key size, Row size and Key location match the Data table.
TB-5940 E	Data Table 'xxxxxxxx' is not on any library searched.	
TB-5941 E	Default Data Table 'xxxxxxxx' is not on any library searched.	
TB-5942 E	You cannot create 'xxxxxxxx'. The table already exists.	
TB-5943 W	Dynamic open failed. View row size 00000 does not match table 00000.	In order to use this View, you must make the Key size, Row size and Key location match the Data table.
TB-5944 E	View 'xxxxxxxx' cannot be found. Enter NEW command to create view.	Enter NEW on the command line to create an View.
TB-5945 W	The table in memory is not the latest generation.	This occurs if another User or application has the table Open for Read in the TSR and you are trying to open the latest copy from the library, that was updated by another application since the table was opened in this TSR.

MSG #	Text	Meaning / Instructions
TB-5946 E	The table in TSR is not the latest generation. Update access denied.	This occurs if another user or application has the table Open for Read in the TSR and you are trying to open the latest copy from the library, that was updated by another application since the table was opened in this TSR. Close the table in the TSR and try the edit again. The tableBASE administrator can go to option 2 on the Administrators Menu to find the User that has the table open.
TB-5947 E	Table is in use by another application. Changes cannot be cancelled.	
TB-5948 W	Help information for menu 'xxxxxxxx' is not found on table 'xxxxxxxx'.	
TB-5950 E	The keys are protected. A new row cannot be inserted in this mode.	
TB-5951 A	This application is now continued at another terminal.	
TB-5952 A	This application is now suspended by the tableBASE administrator.	Contact your tableBASE administrator.
TB-5953 E	View defines Primary Key at 00000 which differs table location 00000.	In order to use this View, you must make the Key size, Row size and Key location match the Data table.
TB-5954 E	View defines Primary Key size of 0000 which differs table key of 0000.	In order to use this View, you must make the Key size, Row size and Key location match the Data table.
TB-5957 W	Line command 'xxxx' at row count ' ' requires target A or B.	
TB-5958 W	Line command 'xxxx' at row count ' ' requires action command.	
TB-5959 E	A hash table organization cannot support a block row delete.	
TB-5960 E	The line commands are conflicting. Row location(s): , .	Examine the conflicting location and place the line command appropriately.
TB-5961 I	Block command 'xxxx' at row count ' ' requires target A or B.	Information only.
TB-5962 I	Block command 'xxxx' at row count ' ' is incomplete.	Information only.
TB-5963 E	Invalid line command 'xxxx', please re-enter.	Re-enter command as required. Typically this occurs if you are using an update line command such as U, N or D while in BROWSE mode.
TB-5964 E	A sequential or hash organization of 'X' cannot support a row move operation. The organization must be U or R. This can be accomplished with a change definition command.	Use option 3 on the Table Define menu to change the table to be user ordered.
TB-5965 I	The number of rows deleted is	Information only.
TB-5966 I	The number of rows moved is x to location y.	Information only.
TB-5967 I	Line commands have been cleared.	Information only.
TB-5968 W	Line commands following the selected row have been deleted.	Under certain organizations multiple row line commands are not supported.

MSG #	Text	Meaning / Instructions
TB-5969 E	A table organization of 'X' cannot support a NEW nn command (insert at location nn operation). The organization must be 'U' (User ordered, Pointer or True) or the organization must be 'R' (Random ordered, True which is changed to User ordered by tablesONLINE). NOTE: A Random Pointer table is edited using a sequential index. In order to change the table organization, use option 3 on the Define Table menu.	
TB-5970 W	Table is empty. Enter data here. To leave table empty select CANCEL.	
TB-5971 E	Line command 'XXXX' invalid in multiple user table access. If you are trying to Delete a row, select the row for Update using the 'U' line command. Once the row is selected, type DELETE in the command line. (DEL or DE will also work.)	Because other users may be editing items, selecting an item first ensures that you have complete control first before deleting the selected row.
TB-5972 E	Multiple line commands are not supported for hash tables.	The rows are physically in random order.
TB-5973 E	tableBASE error 'xxxx' occurred storing table/view 'xxxxxxxx'.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5974 E	Operation suppressed. You are not licensed to extend use of tableBASE.	
TB-5975 E	Editing of VTS server tables is not supported. Try browse.	You accessed a table residing on VTS that was accessed as a result of the VTS name appearing in the tableBASE library list as determined by your profile on the TBOLACT table. Contact your tableBASE administrator.
TB-5980 E	The table could not be opened because the Data Table 'xxxxxxxx' is not a pointer table. The Data Table must be defined as a pointer table in order to be accessed as an alternate indexed table.	This occurs when the underlying data table is not defined as a pointer table. Change the INDEX attribute of the Data table to be pointer.
TB-5985 A	Error updating table 'xxxxxxxx', reason code 9999.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5986 A	Table TBOLACT must be upgraded to Release 6. See Installation Guide.	See Installation Guide for information on updating tablesONLINE/CICS.
TB-5987 A	Table TBOLPROF must be upgraded to Release 6. See Installation Guide.	See Installation Guide for information on updating tablesONLINE/CICS.
TB-5988 A	Table TBOLCNST does not contain the default row, SIGNON terminated.	An important 'DEFAULT ....' row is missing from the TBOLCNST table. This row must exist to launch TBOL for this CICS region. Contact DataKinetics tableBASE support.
TB-5989 A	Table TBOLMRO does not contain the default row, SIGNON terminated.	An important 'DEFAULT ....' row is missing from the TBOLMRO table. This row must exist to launch TBOL for this CICS region. Contact DataKinetics tableBASE support.
TB-5990 A	An error occurred when attempting to terminate your session.	Contact your tableBASE administrator.
TB-5991 A	Unable to open table 'xxxxxxxx', reason code 9999. Job Aborted.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.

MSG #	Text	Meaning / Instructions
TB-5992 A	Transaction work area 99999999 too small. It should be 99999999. Abort	Contact your CICS system administrator.
TB-5993 A	Your application is not defined to the system, SIGNON terminated.	Contact your tableBASE administrator.
TB-5994 A	Descript. '****NO DESCRIPTION ' not on tbl 'xxxxxxxx'	An important '**** NO DESCRIPTION' row is missing from the XXXXDESC table. This row must exist to launch TBOL. Contact DataKinetics tableBASE support.
TB-5995 A	Error creating table 'xxxxxxxx', reason code 9999. Job aborted.	Refer to the tableBASE error return codes for more information. If problem not explained, contact DataKinetics tableBASE support.
TB-5996 A	tablesONLINE incomplete environment.	Check for further tablesONLINE or tableBASE messages.
TB-5997 A	Message '5900 W Help message unavailable, . . . ' not on tbl 'xxxxxxxx'	An important '5900 W Help ....' row is missing from the XXXXMSGS table. This row must exist to launch TBOL. Contact DataKinetics tableBASE support.
TB-5998 W	Unable to save user profile. tBASE COND:	This can happen if the tableBASE administrator has the TBOLPROF table open for edit
TB-5999 A	CICS error: command code 'xx' response code 'xxxxxx'. Job aborted.	Contact your CICS system administrator.
TB-6000 A	The VTS region 'xxxx' needs to be started before launching TBOL.	Contact your tableBASE administrator.
TB-6001 E	The VTS region 'xxxx' needs to be started before 'ttttttt' can be opened. Another approach is to edit the TBSYSVTS table that specifies in which VTS region to open the table.	Contact your tableBASE administrator.
TB-9000 E	Exit program 'xxxxxxxx' called with invalid indicators: xxxxxxxxxxxx.	Contact your tableBASE administrator.
TB-9001 W	Table xxxxxxxx called exit with indicators: xxxxxxxxxxxx in test mode.	Contact your tableBASE administrator.
TB-9999 A	CICS error: command code 'xx' response code 'xxxxxx'. Job aborted.	Contact your CICS system administrator.
TBA1000 E	There is no audit control record for this table. Contact administrator	Contact your tableBASE administrator.

Table E-7:

tablesONLINE/ISPF error messages

The following table contains information on the tablesONLINE/ISPF error messages that can be cross-referenced by the message dataset. Note that the message numbers do not appear on-screen. Use your Browser FIND feature to search for the message text.

Table E-8: tablesONLINE / ISPF messages and error codes

Msg #	Short format	Long format	Meaning
MSG000	TBSYSLB ALLOCATION ERROR	TABLESONLINE/ISPF SYSTEM LIBRARY CANNOT BE ALLOCATED; SEE TBASE ADMIN	Contact system administrator.
MSG001	LIBRARY NOT ALLOCATED	LIBRARY IS ALREADY ALLOCATED OR IS NOT CATALOGUED OR DOES NOT EXIST	
MSG002	TABLE BASE ERROR &TBERR HELP=TBERROR	REFER TO TABLEBASE REFERENCE CARD OR PRESS HELP KEY AGAIN	
MSG003	INVALID GENERATON NUMBER	THE GENERATION NUMBER IS NOT LEFT JUSTIFIED NUMERIC (SIGNED)	
MSG005	TABLE IS EMPTY	TABLE CONTAINS NO ROWS, BROWSE IS SUPPRESSED	
MSG006	NEW TABLE DEFINED	ANEWTABLEHASBEENDEFINEDINSTORAGE	Information only.
MSG007	GENERATION DELETED	THE REQUESTED GENERATION OF TABLE &TABLE HAS BEEN DELETED	Information only.
MSG008	DEFINITION CHANGED	THE DEFINITION OF TABLE &TABLE HAS BEEN CHANGED SUCCESSFULLY	Information only.
MSG009	TABLE SAVED	THETABLEHASBEENSUCCESSFULLYSAVED	Information only.
MSG010	FIELD NAME IS MISSING	THE FIELD NAME IS MISSING FROM THE LINE IDENTIFIED	
MSG011	DISPLAY LEN SUBSTITUTED	THE DISPLAY LENGTH HAS BEEN SUBSTITUTED BY THE VALUE IN THE EDIT TABLE	Information only.
MSG012	DUPLICATE ROW MODIFIED	THIS ROW HAS A KEY THAT IS IDENTICAL TO ANOTHER ONE ON THE TABLE	
MSG013	TABLE IS NOT NEW	A TABLE WITH THIS NAME ALREADY EXISTS ON THIS LIBRARY	
MSG014	KEY FIELDS NOT TOGETHER	THE KEY FIELDS ARE SEPARATED BY NON-KEY FIELDS	
MSG015	TABLE LENGTH SUBSTITUTED	THE TABLE LENGTH HAS BEEN SUBSTITUTED BY THE VALUE IN THE EDIT TABLE	Information only.
MSG016	ROW SIZE CONFLICT	SUM OF FIELD LENGTHS IS NOT EQUAL TO THE DEFINITION ROW SIZE	
MSG017	KEY SIZE CONFLICT	SUM OF KEY FIELD LENGTHS IS NOT EQUAL TO THE DEFINITION KEY SIZE	
MSG018	KEY LOCATION CONFLICT	FIRST KEY FIELD LOCATION IS NOT EQUAL TO THE DEFINITION KEY LOCATION	
MSG019	NO KEY FIELD SPECIFIED	NO KEY FIELD WAS IDENTIFIED AND AT LEAST ONE MUST BE SPECIFIED	
MSG020	TOO MANY KEYS DEFINED	TOO MANY FIELDS HAVE BEEN IDENTIFIED AS KEY FIELDS, MAX 10 FIELDS	

Msg #	Short format	Long format	Meaning
MSG022	VIEW NOT FOUND	A VIEW DEFINITION MUST EXIST FOR THIS FUNCTION.	
MSG023	VIEW EMPTY	NO FIELDS HAVE BEEN DEFINED FOR THIS TABLE.	
MSG024	TOO MANY FIELDS DEFINED	TOO MANY FIELDS HAVE BEEN DEFINED FOR THIS TABLE, SYSTEM LIMITATION	
MSG025	ROW NOT FOUND	THE REQUESTED ROW WAS NOT FOUND IN THE TABLE	
MSG026	SCROLL VALUE INVALID	VALUE MUST BE U (UP), D (DOWN) OR E (EXECUTE) FOLLOWED BY A NUMBER	
MSG027	FORMAT(S) ARE INVALID	THE DISPLAY AND/OR TABLE FORMAT IS INVALID. PLS SEE HELP FOR FORMATS	Refer to HELP or tablesONLINE/ ISPF documentation.
MSG028	TABLE PARMS DO NOT EXIST	THE TRAILER ROW IN THE VIEW DEFINITION DOES NOT EXIST, GO TO OPTION 1	
MSG029	VIEW DEFINITION SAVED	ONLY THE VIEW HAS BEEN SAVED	Information only.
MSG030	COMMAND CONFLICT	INVALID OR ILLEGAL SEQUENCE OF COMMANDS	
MSG031	ROW DELETED	THE ROW SELECTED HAS BEEN DELETED FROM THE TABLE	Information only.
MSG032	ROW ADDED	THE NEW ROW HAS BEEN ADDED TO THE TABLE	Information only.
MSG033	ROW UPDATED	THE ROW SELECTED HAS BEEN UPDATED AS REQUESTED	Information only.
MSG034	DUPLICATE ROW ADDED	THE ROW KEY DUPLICATES AN EXISTING KEY BUT THE ROW WAS ADDED ANYWAY	
MSG035	DISPLAY LEN OUT OF RANGE	THE DISPLAY LENGTH MUST BE BETWEEN &DMINL AND &DMAXL	
MSG036	TABLE LEN OUT OF RANGE	THE TABLE LENGTH MUST BE BETWEEN &TMINL AND TMAXL	
MSG037	TABLE FORMAT CONFLICT	THE TABLE FORMAT SPECIFIED IS IN CONFLICT WITH THE DISPLAY FORMAT	
MSG038	ROW IS EMPTY	TABLE ROW IS EMPTY OR THIS ROW DOES NOT EXIST ON THE TABLE	
MSG039	VIEW NOT FOUND,BASE USED	VIEW WITH DYNAMIC VIEW SUFFIX NOT FOUND, BASE VIEW USED INSTEAD	
MSG040	ALLOCATION ERROR	UNABLE TO ALLOCATE &LIB USING PARAMETERS SPECIFIED	
MSG041	ALLOCATION FAILED	&MLIB, PROBABLY NOT CATALOGUED OR ALREADY ALLOCATED	
MSG042	COPY ERROR - SAME LIB	COPY MUST BE TO ANOTHER LIBRARY IF THE SAME MEMBER NAME IS USED.	
MSG043	ALLOCATION FAILED	ALLOCATION FAILED BECAUSE FROM DATASET OR MEMBER DOES NOT EXIST.	
MSG044	FUNCTION FAILED	INVALID TABLE NAME(S) OR PASSWORD.	Re-enter command, or contact system administrator.
MSG045	DATA IS NOT FORMAT	ERROR &TBERR OCCURRED CONVERTING DISPLAY DATA TO TABLE FORMAT.	
MSG046	RENAME ERROR - SAME NAME	RENAME MUST BE TO DIFFERENT MEMBER NAME.	

Msg #	Short format	Long format	Meaning
MSG047	HOOK PROGRAM NOT LOADED	INSUFFICIENT REGION STORAGE FOR TBHOOK GETMAIN	
MSG048	HOOK PROGRAM MISSING	TBULOAD DATASET NOT FOUND.	
MSG049	HOOK PROGRAM MISSING	TBULOAD DATASET COULD NOT BE OPENED.	
MSG050	DATE IS INVALID	ERROR &TBERR OCCURRED CONVERTING DISPLAY DATA TO TABLE FORMAT	
MSG051	DATA IS NOT NUMERIC	ERROR &TBERR OCCURRED CONVERTING DISPLAY DATA TO TABLE FORMAT	
MSG052	VALUE IS TOO HIGH	ERROR &TBERR OCCURRED CONVERTING DISPLAY DATA TO TABLE FORMAT	
MSG053	USER HOOK ERROR &TBERR	A USER HOOK RETURNED A NON-ZERO RETURN CODE	
MSG054	SERIOUS PROGRAM ERROR	ERROR &TBERR OCCURRED CONVERTING DISPLAY DATA TO TABLE FORMAT	
MSG055	ROW CONTAINS BAD DATA	ERROR &TBERR OCCURRED CONVERTING TABLE DATA TO DISPLAY FORMAT	
MSG056	EXIT PROGRAM MISSING	FIELD DEF SPECIFIES EXITNAME BUT EXIT PGM NOT FOUND IN LOAD LIBRARY	
MSG057	CANT INCREASE ROW SIZE	IF DATA EXISTS IN TABLE, CAN'T INCREASE ROW SIZE VIA TBONLINE	
MSG058	LOCATE FLDNAME NOT FOUND	LOCATE MODE: NO MATCH FOUND ON SUPPLIED LOOKUP KEY	
MSG059	LOCATE FLDNAME -NO KEY	LOCATE MODE: LOOKUP KEY WAS NOT SUPPLIED	
MSG060	VIEW NOT DELETED	A GENERATION OF THE TABLE IS STILL RESIDENT ON THE TABLE LIBRARY	
MSG061	VIEW DELETED	THE VIEW DEFINITION FOR TABLE &TABLE HAS BEEN DELETED	Information only.
MSG062	INSERT BY COUNT FAILED	INSERT BY COUNT CAN ONLY BE PERFORMED ON A TABLE WITH ORGANIZATION = U	
MSG063	COUNT OUTSIDE TABLE	THE COUNT SPECIFIED IS OUTSIDE THE TABLE: LAST ROW COUNT IS &LASTCNT	
MSG064	COUNT SET TO LAST ROW	THE COUNT SPECIFIED IS OUTSIDE THE TABLE: COUNT SET TO LAST ROW	
MSG065	COUNT PAST LAST ROW	THE COUNT SPECIFIED IS OUTSIDE THE TABLE: COUNT SET 1 PAST LAST ROW	
MSG066	COUNT NOT NUMERIC	THE COUNT ENTERED IS BLANK OR CONTAINS INVALID CHARACTERS.	
MSG067	ROW RELOCATED IN TABLE	THE UNCHANGED ROW IS RELOCATED TO ANOTHER LOCATION IN THE TABLE	
MSG068	DUPLICATE KEYS NOT OK	THE TABLE HAS A HASH ORGANIZATION AND DUPLICATE KEYS ARE NOT SUPPORTED	
MSG069	COMMAND UNCLEAR; REENTER	PROBABLY ENTER WAS PRESSED WITH INCOMPLETE DATA IN COMMAND LINE	
MSG070	VIEW NOT FOUND	THE VIEW DEFINITION FOR TABLE &TABLE IS NOT ON THE LIBRARY	
MSG071	THE TABLE IS NOT FOUND	THE TABLE &TABLE IS NOT ON THE LIBRARY	
MSG072	TABLE & VIEW NOT FOUND	NEITHER THE TABLE &TABLE NOR ITS VIEW ARE ON THE LIBRARY	

Msg #	Short format	Long format	Meaning
MSG073	INVALID PASSWORD	TABLE=&TBLNAM IS PASSWORD PROTECTED; PASSWORD SUPPLIED IS INVALID	
MSG074	ROW SIZE IS > 10,000	TABLES WITH ROW-SIZE > 10,000 CAN'T BE ACCESSED VIA TABLESONLINE/ISPF	
MSG075	TABLE EXISTS ON LIBRARY	TABLE = &ALTBL ALREADY EXISTS ON THE LIBRARY	
MSG076	BASE TABLE CANNOT BE ALT	BASE-TABLE CANNOT BE AN ALTERNATE TABLE	
MSG077	ALTERNATE INDEX CREATED	ALTERNATE INDEX HAS BEEN SUCCESSFULLY CREATED	Information only.
MSG078	ALTERNATE INDEX UPDATED	ALTERNATE INDEX HAS BEEN SUCCESSFULLY UPDATED	Information only.
MSG079	BASE TBL MUST BE POINTER	BASE-TABLE MUST BE OF POINTER TYPE INORDER TO CREATE AN ALT DEFINITION	
MSG080	CONVERSION COMPLETE	&CNTRC CONVERTED; &CNTRU UPDATED; &CNTRI NOT CONVERTED	Information only.
MSG081	NO TABLES IN LIBRARY	&LIB, HAS NO ISPF VIEWS IN IT	
MSG082	LIB(FROM) IS MANDATORY	FROM: LIBRARY NAME, IS A MANDATORY INPUT FIELD	
MSG083	INVALID PASSWORD	&CNTRC CONVERTED; &CNTRU UPDATED; &CNTRI NOT CONVERTED	
MSG084	ALLOCATION FAILED	LIBRARY=&MLIB, PROBABLY NOT CATALOGED OR DOES NOT EXIST	
MSG085	NO SPACE ON LIBRARY	&CNTRC CONVERTED; &CNTRU UPDATED; &CNTRI NOT CONVERTED	
MSG086	EXCLUDE=ALL NOT ALLOWED	ALL THE TABLES IN THE LIBRARY CANNOT BE EXCLUDED	
MSG087	TABLE DEFINITION UPDATED	THE DEFINITION OF THE TABLE HAS BEEN UPDATED	
MSG088	NEW VIEW DEFINED	A NEW VIEW HAS BEEN DEFINED	Information only.
MSG089	VIEW UPDATED	THE VIEW HAS BEEN UPDATED	Information only.
MSG090	DSPL AND TBLX LEN NOT EQ	DISPLAYANDTABLELENGTHSMUSTBEEQUAL	
MSG091	D-LEN = T-LEN + 1	DISPLAY LENGTH MUST BE EQUAL TO TABLE LENGTH PLUS ONE	
MSG092	D-LEN = (2 * T-LEN) <-1>	DISPLAY LENGTH MUST BE EQUAL TO TWO TIMES THE TABLE LENGTH (MINUS ONE)	
MSG093	D-LEN = 2 * T-LEN	DISPLAY LENGTH MUST BE EQUAL TO TWO TIMES THE TABLE LENGTH	
MSG094	D-LEN = T-LEN + 1 OR 2	DISPLAY LENGTH MUST BE EQUAL TO TABLE LENGTH PLUS ONE OR PLUS TWO	
MSG095	D-LEN = (2 * T-LEN) <+1>	DISPLAY LENGTH MUST BE EQUAL TO TWO TIMES THE TABLE LENGTH (PLUS ONE)	
MSG096	LENGTH FORMULA INVALID	AN INVALID LENGTH FORMULA HAS BEEN ENCOUNTERED IN THE EDIT TABLE	
MSG097	TBL DEFN CAN'T BE EDITED	TBL DEFN OF AN ALT TBL CAN'T BE EDITED, USE THE CREATE ALT OPTION	
MSG098	SUFFIX MUST BE 1-7 CHAR	THE DYNAMIC SUFFIX INDICATOR MUST BE A 1 TO 7 CHARACTER FIELD	
MSG099	ONLY ONE SUFFIX ALLOWED	IN A FIELD DEFINITION ONLY ONE SUFFIX FIELD IS VALID	

Msg #	Short format	Long format	Meaning
MSG100	CHANGES CANCELLED	CHANGES MADE TO THE TABLE WERE NOT SAVED	Information only.
MSG101	VIEW INCOMPATIBLE (CICS)	VIEW CONTAINS TBOL/CICS R5.0 FEATURES, EDITING MAY CORRUPT TABLE	Information only.
MSG102	VALIDATION ERROR	FIELD CONTAINS LOW VALUES, SPACES OR INVALID DATE	
MSG103	TABLEBASE ONLINE - PRODUCT CODE INVALID	TBONLINE - PROD CODE INVALID	
MSG104	TABLEBASE ONLINE - CUSTOMER CODE MISMATCH	CUSTOMER CODE MISMATCH	
MSG105	TBONLINE-LICENSE EXPIRED	TBONLINE - LICENSE EXPIRED	
MTB000	COMMAND WAS SUCCESSFUL	THE COMMAND WAS PROCESSED SUCCESSFULLY	Information only.
MTB001	INVALID COMMAND	THE SPECIFIED COMMAND IS INVALID	
MTB002	TABLE NOT OPEN	THE SPECIFIED TABLE MUST BE OPEN FOR THIS COMMAND	
MTB003	TABLE IS NOT CLOSED	THE COMMAND REQUIRES THAT THE TABLE BE CLOSED	
MTB004	FG NOT ALLOWED	FG COMMAND IS NOT ALLOWED; TABLE NOT SEQUENTIAL (S OR D)	
MTB005	NO SECONDARY TABLE	THE SEARCH CRITERION WAS NOT FOUND IN THE PRIMARY TABLE	
MTB006	COUNT IS INVALID	THE COUNT SPECIFIED IS INVALID	
MTB007	DSN / DDNAME CHANGED	DSN OR DDNAME HAS BEEN CHANGED. USE DV OR DW BEFORE STORE	Information only.
MTB008	INVALID GENERATION	THE GENERATION NUMBER SPECIFIED FOR THIS TABLE IS INVALID	
MTB009	TABLE NOT FOUND	THE TABLE COULD NOT BE FOUND IN THE LIBRARY(S) SEARCHED	
MTB010	CS PARAMETER INVALID	THE STATUS-SWITCHES PARAMETER HAS INVALID ENTRIES	
MTB011	GENERATIONS INVALID	THE NUMBER OF GENERATIONS TO KEEP IN THE DT MUST BE 1-9	
MTB012	TABLE NAME INVALID	THE TABLE NAME SPECIFIED IS INVALID	
MTB013	COMMAND INVALID	THE COMMAND IS INVALID IN THIS ENVIRONMENT	
MTB014	ROW-SIZE IS INVALID	THE ROW-SIZE SPECIFIED IS INVALID. MUST BE FROM 1-32767	
MTB015	KEY SIZE IS INVALID	THE KEY-SIZE SPECIFIED IS INVALID. MUST BE FROM 1-256	
MTB016	KEY LOCATION IS INVALID	THE KEY-LOCATION SPECIFIED IS INVALID. MUST BE IN THE ROW	
MTB017	KEY NOT WITHIN ROW	THE KEY WILL NOT FIT WITHIN THE ROW	
MTB018	TABLEBASE LIBRARY FULL	INSUFFICIENT SPACE IS AVAILABLE ON THE TABLEBASE LIBRARY	
MTB019	TABLE ALREADY EXISTS	THIS TABLE ALREADY EXISTS ON THE TARGET LIBRARY	

Msg #	Short format	Long format	Meaning
MTB020	MAXNMTAB EXCEEDED	THE MAXIMUM NUMBER OF OPEN TABLES SETTING IS EXCEEDED	
MTB021	ORGANIZATION INVALID	ORGANIZATION SPECIFIED IS INVALID, USE R,U,S,D,H OR BLANK	
MTB022	TABLEBASE ERROR: 22	TABLEBASE ERROR: 22	Call technical support.
MTB023	TABLEBASE ERROR: 23	TABLEBASE ERROR: 23	Call technical support.
MTB024	TABLEBASE ERROR: 24	TABLEBASE ERROR: 24	Call technical support.
MTB025	TABLEBASE ERROR: 25	TABLEBASE ERROR: 25	Call technical support.
MTB026	TABLEBASE ERROR: 26	TABLEBASE ERROR: 26	Call technical support.
MTB027	TABLEBASE ERROR: 27	TABLEBASE ERROR: 27	Call technical support.
MTB028	PAGED NOT ALLOWED	PAGED TABLES ARE NOT SUPPORTED. USE DK1TCNV TO CONVERT	
MTB029	LIBS NOT THE SAME	THE SECONDARY TABLE NOT ON SAME LIBRARY AS PRIMARY TABLE	
MTB030	PASSWORD INVALID	THE PASSWORD SUPPLIED IS INVALID	
MTB031	WRITE PASSWORD INVALID	THE WRITE PASSWORD IS EITHER MISSING OR INCORRECT	
MTB032	TABLE NOT OPENED WRITE	THE ST COMMAND REQUIRES THAT THE TABLE BE OPENED FOR WRITE	
MTB033	DIFFERENT GENERATION	A DIFFERENT GENERATION OF THE TABLE IS ALREADY OPEN	
MTB034	TABLEBASE ERROR: 34	TABLEBASE ERROR: 34	Call technical support.
MTB035	TABLEBASE ERROR: 35	TABLEBASE ERROR: 35	Call technical support.
MTB036	TABLEBASE ERROR: 36	TABLEBASE ERROR: 36	Call technical support.
MTB037	TABLEBASE ERROR: 37	TABLEBASE ERROR: 37	Call technical support.
MTB038	TABLEBASE ERROR: 38	TABLEBASE ERROR: 38	Call technical support.
MTB039	NEW NAME ALREADY EXISTS	RN COMMAND FAILED. THE NEW NAME ALREADY EXISTS ON LIBRARY	
MTB040	DDNAME DOESNT EXIST	THE LIBRARY DDNAME DOES NOT EXIST	
MTB041	SMC VALUE INVALID	THE STORAGE-MODE-CODE (SMC) SPECIFIED MUST BE R OR BLANK	
MTB042	TB-FORMAT INVALID	THE TBPARM SUBPARAMETER TB-FORMAT MUST BE EITHER 0 OR A	
MTB043	ESTIMATED ROWS INVALID	THE ESTIMATED NUMBER-OF-ROWS IN DT BLOCK IS OUT OF RANGE	
MTB044	EXPANSION FACTOR INVALID	THE EXPANSION-FACTOR SPECIFIED IN DT BLOCK MUST BE 1-999	
MTB045	TABLEBASE ERROR: 45	TABLEBASE ERROR: 45	Call technical support.
MTB046	TABLEBASE ERROR: 46	TABLEBASE ERROR: 46	Call technical support.
MTB047	TABLEBASE ERROR: 47	TABLEBASE ERROR: 47	Call technical support.
MTB048	TABLEBASE ERROR: 48	TABLEBASE ERROR: 48	Call technical support.
MTB049	TOO FEW PARAMETERS	THIS COMMAND REQUIRES MORE PARAMETERS THAN WERE GIVEN	
MTB050	DIRTYPE INVALID	DIRTYPE SPECIFIED FOR THE LD COMMAND MUST BE V,D,T, BLANK	
MTB051	DENSITY INVALID	THE DENSITY SPECIFIED IN THE DT BLOCK MUST BE FROM 1-999	
MTB052	TABLEBASE ERROR: 52	TABLEBASE ERROR: 52	Call technical support.

Msg #	Short format	Long format	Meaning
MTB053	TABLEBASE ERROR: 53	TABLEBASE ERROR: 53	Call technical support.
MTB054	TABLEBASE ERROR: 54	TABLEBASE ERROR: 54	Call technical support.
MTB055	SEARCH-METHOD INVALID	THE SEARCH-METHOD MUST BE EITHER S,Q,B,C,H	
MTB056	INCOMPATIBLE METH/ORG	SEARCH-METHOD IS INCOMPATIBLE WITH ORGANIZATION SPECIFIED	
MTB057	TABLEBASE ERROR: 57	TABLEBASE ERROR: 57	Call technical support.
MTB058	MODULE NOT FOUND	THE REQUESTED MODULE CANNOT BE FOUND IN THE LOADLIB	
MTB059	TABLEBASE ERROR: 59	TABLEBASE ERROR: 59	Call technical support.
MTB060	LIBRARY INVALID	THE SPECIFIED LIBRARY IS NOT SUITABLE	
MTB061	LIBRARY STATUS INVALID	THE LIBRARY STATUS IS INVALID SEE ERROR MSG 61 SUBCODES	
MTB062	LIBRARY FORMAT ERR	FORMAT OF SPECIFIED LIBRARY INCOMPATIBLE WITH VERSION 6	
MTB063	TABLEBASE ERROR: 63	TABLEBASE ERROR: 63	Call technical support.
MTB064	INVALID INDEX PARAMETER	THE INDEX PARAMETER SPECIFIED MUST BE P, T OR BLANK	
MTB065	TABLEBASE ERROR: 65	TABLEBASE ERROR: 65	Call technical support.
MTB066	TABLEBASE ERROR: 66	TABLEBASE ERROR: 66	Call technical support.
MTB067	TABLEBASE ERROR: 67	TABLEBASE ERROR: 67	Call technical support.
MTB068	COUNT TOO SMALL ON DU	THE COUNT VALUE IS TOO SMALL FOR DU COMMAND TO DUMP ROWS	
MTB069	TABLEBASE ERROR: 69	TABLEBASE ERROR: 69	Call technical support.
MTB070	TABLEBASE ERROR: 70	TABLEBASE ERROR: 70	Call technical support.
MTB071	LOCK WAIT EXCEEDED	WAIT FOR LOCK EXCEEDED LOCKTIMERC VALUE	
MTB072	TABLE UNAVAILABLE	THE TABLE IS UNAVAILABLE AT THIS TIME, WAIT=N IS IN EFFECT	
MTB073	NO UPDATING ALLOWED	THIS TABLE IS OPEN FOR READ, NO UPDATING ALLOWED	
MTB074	LOCK-LATCH INVALID	THE LOCK-LATCH SPECIFIED IS INVALID	
MTB075	ALLOC/DEALLOC FAILED	DYNAMIC ALLOCATION (AL) OR UN-ALLOCATION (UL) FAILED	
MTB076	PRIMARY TABLE ERROR	OPEN INDIRECT FAILED, PRIMARY NOT SEQUENTIAL OR KLOC NOT 9	
MTB077	TABLE NOT POINTER	ALTERNATE INDEX DEFINED FOR A TABLE THAT IS NOT A POINTER	
MTB078	ALLOC DISP INVALID	AL COMMAND FAILED, SHARE-STATUS PARAMETER MUST BE S OR O	
MTB079	TABLEBASE ERROR: 79	TABLEBASE ERROR: 79	Call technical support.
MTB080	DATA TABLE NOT OPEN	DATA TABLE NOT OPEN. USE OF ALTERNATE INDEX NOT ALLOWED	
MTB081	ALT DEF NOT FOUND	THE SPECIFIED ALTERNATE INDEX DEFINITION IS NOT FOUND	
MTB082	TABLEBASE ERROR: 82	TABLEBASE ERROR: 82	Call technical support.
MTB083	TABLE NOT POINTER	THE DATA TABLE INDEX SUBPARAMETER OF DT BLOCK MUST BE P	

Msg #	Short format	Long format	Meaning
MTB084	TABLEBASE ERROR: 84	TABLEBASE ERROR: 84	Call technical support.
MTB085	INVALID ALT COMMAND	THE COMMAND IS NOT VALID FOR ALTERNATE TABLE	
MTB086	OPEN INDEX FAILED	OPEN OF THE INDEX FAILED ON A LINKED TABLE IN A VTS-TSR	
MTB087	KEY-COUNT MUST BE 1	THE KEY-COUNT MUST BE 1 FOR DEFINITION OF ALTERNATE INDEX	
MTB088	TABLEBASE ERROR: 88	TABLEBASE ERROR: 88	Call technical support.
MTB089	ERROR IN TBOPT CARD	AN INVALID PARAMETER WAS ENCOUNTERED IN THE TBOPT FILE	
MTB090	INSUFFICIENT MEMORY	INSUFFICIENT MAIN STORAGE AVAILABLE. INCREASE REGION SIZE	
MTB091	I/O ERROR	I/O ERROR WHILE ATTEMPTING TO READ/ WRITE TO LIBRARY/FILE	
MTB092	INSUFF TSR SIZE	INSUFFICIENT TABLE SPACE REGION (TSR) SIZE	
MTB093	RF INCOMPLETE	THE RF COMMAND DID NOT COMPLETE	
MTB094	TABLEBASE ERROR: 94	TABLEBASE ERROR: 94	Call technical support.
MTB095	TRANSACTION FAILED	THE TRANSACTION (JOB STEP) FAILED	
MTB096	TABLEBASE ERROR: 96	TABLEBASE ERROR: 96	Call technical support.
MTB097	ML/LL LIST INVALID	THE ML OR LL PARAMETER LIST CONTAINS A REFERENCE TO A VTS	
MTB098	INSTALL PARMS INVALID	THE INSTALLATION PARMS ARE NOT VALID FOR THIS OPERATION	
MTB099	TABLEBASE ERROR: 99	TABLEBASE ERROR: 99	Call technical support.

Other error messages

Error and audit messages generated by the batch utility program TBEXEC, and other programs are identified and described in alphabetical order in the following table.

Messages issued from tableBASE in conjunction with the TBEXEC and other programs/ utilities vary from function to function, and are listed in the tables below. The error code is shown at the end of the message, while the originating program can be identified by the message prefix. The following prefixes may be encountered:

IOSRT-51— indicates TBEXEC in conjunction with subroutine DK1T0051
T1182— indicates utility DK1TCNV in conjunction with subroutine DK1T0051

Table E-9: Messages and error codes for TBEXEC: DK1T0051

Msg Code	Meaning / Instructions
IOSRT-51 <message> ERROR 22	DDNAME provided in command is not a valid DDNAME or it is not allocated in the jobstream.
IOSRT-51 <message> ERROR 27	The dataset for the DDname specified must be a DISK file; it is not.
IOSRT-51 <message> ERROR 30	The DSORG (dataset organization) for the DDNAME specified must be compatible for the usage: VSAM RRDS, BDAM or QSAM for a library, QSAM or a member of a PDS(E), SYSIN or SYSOUT for a sequential dataset.
IOSRT-51 <message> ERROR 31	The RECFM (Record Format) for the DDNAME specified must be F for a library, F or FB for sequential input. FA and FBA are allowed for sequential output of Report files.
IOSRT-51 <message> ERROR 32	The BLKSIZE (Block Size) parameter for the DDNAME specified must be a multiple of record length.
IOSRT-51 <message> ERROR 33	The output dataset for this DDname may not be allocated DISP=SHR.
IOSRT-51 <message> ERROR 34	An error was encountered processing a VSAM dataset. See the JESMSGLG for a description of the VSAM error.

Note: The message handling is used by many processes. Depending upon how the message is generated, it may contain only the IOSRT error message number and the associated message text (shown as <message> above), or just the message number.

Table E-10:

Table E-11: Messages and error codes for library conversion

Msg Code	Meaning / Instructions
T1182: <message> ERROR Invalid DDname	DDNAME provided in command is not a valid DDNAME or it is not allocated in the jobstream.
T1182: <message> ERROR Not a disk file	The dataset for the DDname specified must be a DISK file; it is not.
T1182: <message> ERROR Non zero parameters	The DSORG (dataset organization) for the DDNAME specified must be compatible for the usage: VSAM RRDS, BDAM or QSAM for a library, QSAM or a member of a PDS(E), SYSIN or SYSOUT for a sequential dataset.
T1182: <message> ERROR RECFM	The RECFM (Record Format) for the DDNAME specified must be F for a library, F or FB for sequential input. FA and FBA are allowed for sequential output of Report files.
T1182: <message> ERROR BKLSIZE	The BLKSIZE (Block Size) parameter for the DDNAME specified must be a multiple of record length.
T1182: <message> ERROR SHR WRITE	The output dataset for this DDname may not be allocated DISP=SHR.
T1182: <message> ERROR VSAM	An error was encountered processing a VSAM dataset. See the JESMSGLG for a description of the VSAM error.
T1182: <message> INVALID BLKSIZE	The specified blocksize for the target library was invalid. tableBASE library blocksizes must be 3120.

Note: Certain structural errors in page tables on Version 5 tableBASE libraries may cause the Library conversion process to fail with abend S209. This same error can occur when running DK1TLCHK.

Table E-12:

