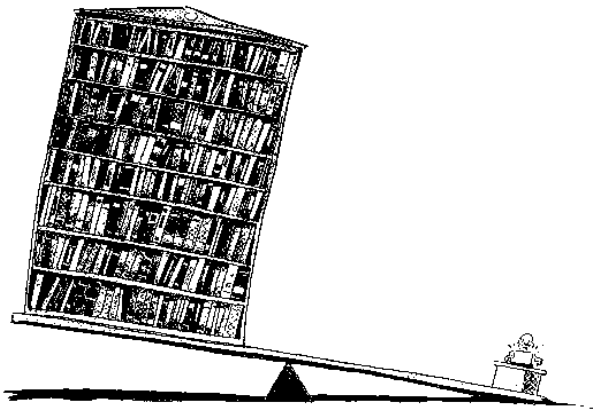
KE Texpress



Client/Server API Guide

KE Software Pty Ltd

Contents

Chapter 1

Introduction

Overview

The KE Texpress Information Management System is an object-oriented database package which provides numerous extensions to the traditional relational database model. The most significant extensions are in the area of complex object support. KE Texpress supports the inclusion of the following object components into an object definition:

- Text.
- Multi-valued fields
- References to foreign objects (objects in different formats controlled by software other than KE Texpress).

This manual describes the KE Texpress C Language Applications Programming Interface (C-API). The C-API provides a back-end library of C functions which enable developers to harness the speed and flexibility of the KE Texpress Information Management System.

The library of functions provides a wrapper around the Texql language so that the full functionality of Texql is available through the API. The KE Texpress C-API may also be accessed using the C++ programming language. For a complete description of Texql refer to the Texql Guide.

The C-API can also be used in conjunction with Titan 3.4 databases.

Function descriptions in this manual provide formal C code definitions for the function call, parameters passed and values returned. Typically an example section of code which utilises the function is also shown.

The remainder of this manual is divided into the following chapters.

Chapter 2 describes the method by which function error codes and messages can be accessed.

In chapter 3 the API initialisation and termination functions are discussed.

Chapter 4 provides an overview of the primary API functions used for performing commands. Chapters 5 and 6 describe the row access and column access functions respectively.

Convenience functions are described in Chapter 7. These functions provide short hand methods of accessing data in a style particular suited to the data layout of Titan 3.4 databases.

A complete example program is provided in Chapter 8.

The sample code sections and the complete example program utilise the *contacts*, *loantypes* and *loans* tables described in the Texql Guide.

Terminology

KE Texpress uses terminology which reflects the object-oriented nature of the product, and thus highlights the distinction between it and relational database systems. However, Texql provides an interface to KE Texpress databases which attempts is similar to a standard SQL interface to a relational database.

This section describes the terminology used by Texql and the C-API in terms of the appropriate terminology of KE Texpress. Refer to the KE Texpress Guides for a description of KE Texpress terminology.

The following terms are used throughout this manual:

Texql	**KE Texpress**
table	This refers to a single KE Texpress database. All KE Texpress databases, although controlled separately in terms of access privileges, etc., are accessible as Texql tables.
column	This refers to an item in a KE Texpress database.
nested table	This refers to a KE Texpress multi-field item which is not of type text or a multi-field text item without an associated Look-up table. Multi-field text items without Look-up tables are considered to be Texql text boxes, i.e. single atomic value of (continuous) text.
tuple or **row**	This refers to a record in a KE Texpress table or a record derived by Texql as the result of a query.
nested tuple	The multi-field Key and library items of KE Texpress are represented as nested tuples in Texql. This means that these items can be treated as atomic values or, alternatively, their components can be individually manipulated.
atomic value	This refers to a value in a column of a tuple, i.e. the value of a field within a KE Texpress record

To assist in the portability of API client programs between various platforms C language typedefs are used for function arguments (e.g. TEXCURSOR, TEXS32, TEXSTRING). Refer to the API C language header files (the "include" directory) for further information.

Compiling an Application Program

Application programs which use the KE Texpress C-API need access to the API header file and the API library. These files are kept under the KE Texpress directory. On a UNIX system, for example, if the KE Texpress home directory is:

```
/home/kestrel/ texpress
```

the API related information resides in the following directories:

```
/home/kestrel/ texpress/include
/home/kestrel/ texpress/lib
```

All C source files which use the KE Texpress C-API function calls must include the KE Texpress API header file. This file is included by using the compiler directive:

```
#include         "texapi.h"
```

or

```
#include         <texapi.h>
```

To ensure maximum portability of applications the first form is preferred. The include file is located in the *include* directory under the API directory. On a UNIX system, this directory is typically specified on the C compiler command line as one to search for header files.

The API libraries reside in the *lib* directory under the API directory. On a UNIX system, this directory is typically specifed on the C compiler or loader line as one to search for library files.

Thus to compile an application program in the C source code file *prog.c* into an executable program *prog* on a UNIX system the following command can be used:

```
cc  -I/home/kestrel/ texpress/include   prog.c  \
    -L/home/kestrel/ texpress/lib  \
    - ltex  - los  \
    -o  prog
```

Alternatively, an environment variable (or a macro in a Makefile) can be set to point to the texpress directory:

```
setenv TEXAPI /home/kestrel/ texpress
cc  -I${TEXAPI}/include  prog.c  \
    -L${TEXAPI}/lib  - ltex  - los  -o  prog
```

Chapter 2

Error Handling

Overview

All KE Texpress API functions report errors in a consistent manner. Each function returns a status value of 0, indicating success, or -1, indicating an error. If an error status is returned then the error function described in this chapter may be used to obtain information about the type of error that has occurred.

It is considered good programming practice to always check the return value of an API function call.

Error Number

NAME

TexError - error number

SYNOPSIS

```
int
TexError()
```

DESCRIPTION

Gets the error number of an error generated by the last API call. A full list of error numbers is contained in the"texapi.h" header file.

RETURN VALUES

The error number.

ERRORS

None

EXAMPLE

```
...

printf("API call failed: no. = % d\n", TexError());
...
```

SEE ALSO

TexErrMsg, TexErrOff

Error Message

NAME

TexErrMsg - error message

SYNOPSIS

```
TEXSTRING
TexErrMsg()
```

DESCRIPTION

Gets the error message of the error generated by the last API call. The text of these error messages is kept in the standard KE Texpress text file.

RETURN VALUES

A pointer to the text of the error message.

ERRORS

None

EXAMPLE

```
...
printf("API call failed: \"% s\"\n", TexErrMsg());
...
```

SEE ALSO

TexError

Error Offset in Command

NAME

TexErrOff - offset of error in texql statement text

SYNOPSIS

```
int
TexErrOff()
```

DESCRIPTION

Gets the character offset in the texql statement text of the last error generated by a call to the KE Texpress API. If the error was not directly associated with a texql statement, this value is -1.

RETURN VALUES

The character offset.
-1 if error not associated with texql statement.

ERRORS

None

EXAMPLE

```
TEXCURSOR       cursor
char            cmd[128];
int             off;
...
sprintf( cmd, "select all  frim contacts");
if (TexCommand( cmd, &cursor) < 0)
{
        switch ( TexError())
        {
            case  TESYNTAX:
                off = TexErrOff();
                printf("Syntax error  in\n%s\n",  cmd);
                printf("Near offset % d\n", off);
                break;
                ...
}
...
```

SEE ALSO

TexError

Chapter 3

Initialisation and Termination

Overview

Before using other functions, each program must first initialise the KE Texpress API and then connect to at least one KE Texpress server. A termination call is also provided for when access to the API is no longer required.

If a connection is established correctly a session identifier is returned. This session identifier is used in subsequent calls to indicate which server is being accessed.

The version number of the current API installation may be obtained to allow the calling program to check for compatibility.

The KE Texpress API make extensive use of KE Texpress tables. The TexTable() function allows the calling program to determine the availibility of a KE Texpress table at programstartup.

Connection Parameters

NAME

TEXPARAMS - structure holding parameters used when establishing API connections.

SYNOPSIS

```
#include        "texapi.h"
```

DESCRIPTION

The connection function TexConnect is passed parameters which define the connection to be established in a TEXPARAMS structure. The stucture holds the following entries:

```
TEXSTRING       p_name;
int             p_type;
TEXSTRING       p_host;
TEXSTRING       p_port;
TEXSTRING       p_service;
TEXSTRING       p_prog;
TEXS32          p_read;
TEXS32          p_write;
TEXSTRING       p_user;
TEXSTRING       p_passwd;
char            p_escape;
TEXS32          p_baud;
int             p_parity;
int             p_stop;
```

The *p_name* field may hold a name which can be used to refer to the connection parameters. It is currently unused by the API.

The *p_type* field holds a flag indicating the type of connection to be established. The value in this field should be one of the pre-defined constants:

```
IO_SOCKET
IO_PIPE
IO_SERIAL
```

The *p_host* field holds the name of the host machine to connect to when establishing a socket connection. The *p_port* field holds the name of the serial port to use when establishing a serial connection. Note that the use of these two fields is mutually exclusive.

The *p_service* field holds the name of the service to connect to when establishing a socket connection. This name should appear in the file /etc/services when establishing socket connections from UNIX machines.

The *p_prog* holds the name of the program to run when establishing using a pipe to communicate with the server. Note that the use of these two fields is mutually exclusive.

The *p_read* field holds the size of buffer to use for receiving data from the server. The *p_write* field holds the size of buffer to use when transmitting data to the server.

The *p_user* field holds the login name of the user to run as on the server machine. This user's privileges will control the degree of access provided to the tables used by the API. The *p_passwd* field holds the unencrypted password of this user. These values are transmitted to the server during the TexConnect call to authenticate access to the server machine. The values are not used for pipe-based connections. A NULL password may be sent for connections on a socket. This will force the server to use the remote command authentication scheme based on the UNIX hosts.equiv and .rhosts files This is the mechanism used by the standard UNIX remote command utilities rlogin, rsh, and rcp. (For more information, see the hosts.equiv entry in section 4 of the UNIX manuals).

The remaining fields are only used when establishing serial line connections to the server.

Serial-based connections use XON/XOFF flow control to prevent the loss of information during transmission of large amounts of data. The *p_escape* field holds the character to be used to escape the special meaning of certain characters (primarily the XON (control-Q) and XOFF (control-S) characters themselves) during serial transmission.

The *p_baud* field holds the data transmission rate to be used. A set of pre-defined constants is supplied to specify the baud rate:

```
IO_BAUD_300
IO_BAUD_1200
IO_BAUD_9600
...
```

etc. Numeric values may also be specified. Not all baud rates may be supported by all connections. When a baud rate is requested, the nearest slower rate supported by both client and server machines is selected.

The *p_parity* field holds the parity to use when transmitting each character. The value in this field should be one of the pre-defined constants:

```
IO_PARITY_NONE
IO_PARITY_EVEN
IO_PARITY_ODD
```

When selecting which parity to use, it is important to realise that all data transmission must be with 8-bit characters. Unless extended parity generation is supported (i.e. a ninth parity bit is transmitted for each character) *IO_PARITY_NONE* should be used.

The *p_stop* field holds the number of stop bits to transmit between each character. The value in this field should be one of the pre-defined constants:

```
IO_STOP_10
IO_STOP_20
IO_STOP_15
```

which represent using 1, 2 and 1.5 stop bits respectively. Not all of these values will be supported by all clients and servers

DEFAULT VALUES

p_type	*IO_PIPE* for UNIX clients *IO_SERIAL* for DOS, Windows and Macintosh clients.
p_host	*"localhost"*
p_port	*"COMM1"* for DOS and Windows clients. *"modem"* for Macintosh clients.
p_service	*"texserver"*
p_prog	*"texserver"*
p_read	*1024*
p_write	*1024*
p_user	The login name of the effective user on the client machine for UNIX clients. The host name (if any) for DOS, Windows and Macintosh clients.
p_passwd	*NULL*
p_escape	*ESC* (Octal 033)
p_baud	*IO_BAUD_9600*
p_parity	*IO_PARITY_NONE*
p_stop	*IO_NONE*

SEE ALSO

TexInitialise,TexConnect,TexParams

API Initialisation

NAME

TexInitialise - initialise the API.

SYNOPSIS

```
int
TexInitialise( argc, argv, params)
int             *argc;
char            **argv;
TEXPARAMS       *params;
```

DESCRIPTION

Performs the necessary initialisation for the front-end of the KE Texpress API. No other API functions should be called prior to this function. Command line arguments should be passed through to TexInitialise. A pointer to a TEXPARAMS structure is also passed. This structure is loaded with default connection parameters, based on the client machine-type, the values of certain environment variables and the values specified by any API-specific command line arguments. Any arguments specific to the API are removed from the argument list.

The environment variables which are interpreted by TexInitialise are:

Environment Variable	TEXPARAMS field affected	Value
TEXTYPE	p_type	"socket", "pipe", or "serial"
TEXHOST	p_host	Host for a socket connection.
TEXPORT	p_port	Port for a serial connection
TEXSERVICE	p_service	Service for a socket connection
TEXPROG	p_prog	Program for a pipe connection
TEXREAD	p_read	Receive buffer size in bytes
TEXWRITE	p_write	Transmit buffer size in bytes
TEXUSER	p_user	User login name on server machine
TEXESCAPE	p_escape	Escape character
TEXBAUD	p_baud	Data transmission rate
TEXPARITY	p_parity	"none", "even", or "odd"
TEXSTOP	p_stop	"1", "2", or "1.5"

The arguments which the TexInitialise consumes are all introduced on the command line by '-T'. The options which may be specified are:

Command-line Argument	TEXPARAMS field affected	Meaning
-Tb*size*	p_read p_write	Receive and transmit with *size* buffer size
-Tc	p_type	Use a serial connection.
-Th*host*	p_host	Connect to *host* for socket connection
-Tn	p_type	Use a socket connection
-Tp	p_type	Use a pipe connection
-Tr*size*	p_read	Receive with *size* buffer size
-Ts*service*	p_service	Use *service* for socket connection
-Tw*size*	p_write	Transmit with *size* buffer size

RETURN VALUES

0	API front-end was successfully initialised.
-1	Initialisation procedure failed.

ERRORS

None.

EXAMPLE

```
main(argc, argv)
int     argc;
char    **argv;
{
        TEXPARAMS       params;
...
        if (TexInitialise(&argc, argv, &params) < 0)
                /* initialisation failure */
...
```

SEE ALSO

TexConnect, TexParams, TexDisconnect, TexTerminate

Server Connection

NAME

TexConnect - connect to a KE Texpress server.

SYNOPSIS

```
int
TexConnect( params, session)
TEXPARAMS       *params;
TEXSESSION      *session;
```

DESCRIPTION

Connects to a KE Texpress back-end server. The connection is established according to the configuration held in the structure pointed to by the *params* argument. The connection invokes the server on the host machine. If the connection is successfully established then a session identifier is returned in the *session* parameter. The session identifier is used in subsequent API calls.

More than one call to TexConnect can be made. This means that more than one server, running on more than one host machine, can be invoked by the one API application program.

RETURN VALUES

0	Server connection was successfully established.
-1	Connection procedure failed.

ERRORS

TELICENCEERR	Licencing error.
TEWHOAREYOU	User information could not be determined.
TEPERMISSION	No permission to connect to this server.

EXAMPLE

```
TEXPARAMS       params;
TEXSESSION      session;
...
if (TexConnect(& params, &session) < 0)
        /* connection failure */
...
```

SEE ALSO

TexInitialise,TexParams,TexDisconnect,TexTerminate

Session Parameters

NAME

TexParams - get current session parameters

SYNOPSIS

```
int
TexParams(session,  params)
TEXSESSION       session;
TEXPARAMS        *params;
```

DESCRIPTION

Retrieves the current connection parameters for the session whose identifier is passed in the argument *session*. The *params* argument points to a TEXPARAMS structure which is loaded with the session's connection parameters. These parameters may be used to check that the appropriate connection has been made.

RETURN VALUES

0	Parameters were retrieved successfully.
-1	Parameters could not be retrieved.

ERRORS

TESESSIONBAD	An incorrect session identifier was supplied
TESESSIONCLOSED	The connection is no longer open.

EXAMPLE

```
TEXSESSION       session;
TEXPARAMS        params;
...
if (TexParams(session, & params) < 0)
        /* connection failure */
...
printf("connection host is % s\n", params.p_host);
...
```

SEE ALSO

TexInitialise,TexConnect,TexDisconnect,TexTerminate

API Version

NAME

TexVersion - determine API version

SYNOPSIS

```
int
TexVersion(session, version)
TEXSESSION       session;
TEXSTRING        *version;
```

DESCRIPTION

Retrieves the version number of the server running on the machine with the session identifier *session*. This function can useful for providing version release verfication for front end applications which use the API.

RETURN VALUES

0 Version numbersuccesfully determined.
-1 Error in determining the version number.

ERRORS

TESESSIONBAD	An incorrect session identifier was supplied
TESESSIONCLOSED	The connection is no longer open.

EXAMPLE

```
TEXSTRING        version;
...
TexVersion(session, &version);
if (strcmp(version, "5.0.12") != 0)
{
        printf("Program requires version 5.0.12\n");
        exit(1);
}
```

SEE ALSO

TexInitialise,TexConnect,TexDisconnect

Table Access

NAME

TexTable - open a table for access

SYNOPSIS

```
int
TexTable(session, table)
TEXSESSION      session;
TEXSTRING       table;
```

DESCRIPTION

Opens a KE Texpress table on the machine whose session identifier is *session* for access by subsequent API calls. By default tables are opened when first referenced in any session. Tables remain open until the session is terminated.

This function may be used to explicitly open a table so as to provide more specialised error diagnostics.

RETURN VALUES

0	Table was opened successfully.
-1	Table could not be opened.

ERRORS

TESESSIONBAD	An incorrect session identifier was supplied
TESESSIONCLOSED	The connection is no longer open.
TETABLEFAIL	No such KE Texpress table.
TENOREG	User is not a registered user of the table.
TENOINIT	Table is not initialised.
TELOGON	Unable to use the table at this time.
TETABLESTART	General table start up error.
TETABLEREAD	Unable to read table description.
TETABLENOISE	Table noise word file start up error.

EXAMPLE

```
if (TexTable(session, "loans") < 0)
{
        if (TexError() ==  TETABLEFAIL)
                printf("loans table not  found\n");
        ...
}
```

SEE ALSO

TexCommand,TexInitialise,TexTerminate

Server Configuration

NAME

TexConfQuote - set the text quoting character

SYNOPSIS

```
int
TexConfQuote(session, quote)
TEXSESSION      session;
char            *quote;
```

DESCRIPTION

Sets the text quoting character to the character pointed to by the *quote* argument for the server running on the machine identified by *session*.

This function is used to modify the text quoting character. If the character pointed to by the *quote* argument is null (the '\0' character) the default quote character is used (usually the single quote, '\''). The value pointed to by the *quote* argument is filled with the character which is used.

RETURN VALUES

0	Server was configured.
-1	Server could not be configured.

ERRORS

TESESSIONBAD	An incorrect session identifier was supplied
TESESSIONCLOSED	The connection is no longer open.

EXAMPLE

```
TEXSESSION      session;
TEXCURSOR       cursor;
char            chr, *cmd;
...
chr = ':';
if (TexConfQuote(Session, & chr) < 0)
        /* handle error */
cmd = "contacts where surname = :  o'connor:";
if (TexCommand(session,  cmd, &cursor) < 0)
        /* handle error */
```

SEE ALSO

TexError, TexCommand,TexInitialise,TexTerminate

Server Interruption

NAME

TexInterrupt - interrupt a server function

SYNOPSIS

```
int
TexInterrupt(session)
TEXSESSION      session;
```

DESCRIPTION

Interrupts the function which is executing on the machine whose session identifier is*session*.

This function is typically called by a signal handler which has responded to the user interrupting the front-end process during the execution of another API operation. The function which is interrupted will return with an error status and a subsequent call to TexError() will return the result code TEINTERRUPT.

RETURN VALUES

0	Server was interrupted.
-1	Server could not be interrupted.

ERRORS

TESESSIONBAD	An incorrect session identifier was supplied
TESESSIONCLOSED	The connection is no longer open.

EXAMPLE

```
TEXSESSION      Session;
...
extern void     handler( int);
...
signal( SIGINT, handler);
...
if ( TexCommand(Session, "contacts where remarks \
                                    contains ' f*'") < 0)
{
        switch ( TexError())
        {
            case  TEINTERRUPT:
                printf(" TexCommand was  interrupted\n");
                break;
        }
        ...
```

```
}
...
void
handler( sig)
int      sig;
{
         /* User interrupted front-end so request any
         ** back-end o peration is terminated.
         */
         TexInterrupt(Session);
}
```

SEE ALSO

TexError, TexCommand,TexInitialise,TexTerminate

Server Disconnection

NAME

TexDisconnect - close an established connection

SYNOPSIS

```
int
TexDisconnect(session)
TEXSESSION      session;
```

DESCRIPTION

Closes a previously established connection to a KE Texpress server. The session identifier is given in the *session* argument. All open cursors associated with the session are closed.

RETURN VALUES

0	API was successfully terminated.
-1	The terminate request failed.

ERRORS

TESESSIONBAD	An incorrect session identifier was supplied
TESESSIONCLOSED	The connection is no longer open.

EXAMPLE

```
main(argc, argv)
int     argc;
char    **argv
{
...
        if (TexConnect(&params, &session) < 0)
                /* connection failure */
...
        TexDisconnect(session);
}
```

SEE ALSO

TexInitialise,TexConnect,TexParams

API Termination

NAME

TexTerminate - terminate the API

SYNOPSIS

```
int
TexTerminate()
```

DESCRIPTION

Terminates the KE Texpress API. No other calls should be made to API functions after this function is called. This call shutdowns all open connections thereby closing all previously accessed tables. All memory used by the API is freed.

RETURN VALUES

0	API was successfully terminated.
-1	The terminate request failed.

ERRORS

None

EXAMPLE

```
main(argc, argv)
int     argc;
char    **argv
{
        TEXPARAMS       params;
...
        if (TexInitialise(&argc, argv, &params) < 0)
                /* initialisation failure */
...
        TexTerminate();
        exit(0);
}
```

SEE ALSO

TexInitialise,TexConnect

Chapter 4

Cursors

Overview

All access to tables is initially via a Texql statement sent to the API. Any valid Texql command may be sent to the API for processing. When the command succesfully completes a cursor is assigned and returned to the calling function. This cursor is used for subsequent column and data access.

Two groups of functions are provided for use with a cursor. The TexCol() group of functions can be used to access the column structure for the operation as well as obtain an actual column data value. The TexRow() group of commands can be used to manipulate the row markers which indicate the next area of data to be retrieved.

It is perfectly acceptable to have multiple cursors in operation at the same time. When a cursor is no longer required it should be closed.

A cursor may refer to the result of any Texql statement. This means that a cursor may refer to a table, a tuple or an atom. Generally a cursor is used to manipulate the rows and columns of a table. However, nested cursors (see Column Nested Cursor) may be used to access the nested columns and rows of nested tables or the columns of a nested tuple.

A cursor is associated with a specific connection. The connection's session identifier is stored with the cursor and so need not be passed to subsequent API calls which use the cursor.

Texql Command

NAME

TexCommand - perform a Texql command

SYNOPSIS

```
int
TexCommand(session, command, cursor)
TEXSESSION      session;
TEXSTRING       command;
TEXCURSOR       *cursor;
```

DESCRIPTION

Performs the requested command on the server identifer by the *session* argument. The *command* argument is interpreted as a texql command. If the command completes successfully then a Texql cursor is opened and assigned to the *cursor* parameter. This cursor is then used to access data and other information associated with the command.

For Texql query commands the cursor can be used in subsequent API calls to retrieve the row and column data that matched the query. Initially the row marker is placed at the first row that matched the query. The TexRowNext() and TexRowGet() functions may be used as required to manipulate the row marker. The TexCol() group of functions are used to actually access the data.

For Texql describe commands the cursor can be used to retrieve the resulting column structure for the command. The TexCol() group of functions are used to access the column structure. As no actual data is associated with a describe cursor the TexColDataGet() and TexColDataSet() functions and the TexRow() group of functions are invalid in this instance.

For Texql data manipulation (DML) commands the cursor is made available so the number of rows manipulated can be determined using TexRowCnt(). No other operations (other than TexClose()) are valid on DML cursors.

RETURN VALUES

0	The command succeeded.
-1	The command failed.

ERRORS

TECURNOMORENo more cursors are available.

TESYNTAX Syntax error in command.

Many other error codes are possible. Refer to "texapi.h" for a complete list of error codes. Also refer to the Texql Guide for a complete description of the Texql language and of the error codes generated.

EXAMPLES

```
TEXSESSION      session;
TEXCURSOR       cursor;
...
if (TexCommand(session, "select all from contacts",
                                                &cursor) < 0)
        /* Texql command failed - check error */
        ...

TEXSESSION      session;
char            *cntry, cmd[128];
int             maxexp;
TEXCURSOR       cursor;
..
cntry = "Japan";
maxexp = 300000;
...
sprintf(cmd, "contacts where country = '%s'
                and exposure <= %d", cntry, maxexp);
if (TexCommand(session, cmd, &cursor) < 0)
        /* check error */
        ...
else
        /* can now process matching values */
        ...

TEXSESSION      session;
char            table[30], cmd[128];
TEXCURSOR       cursor;
...
printf("Table to describe? ");
if (gets(table))
{
        ...
        sprintf(cmd, "describe %s", table);
        if (TexCommand(session, cmd, &cursor) < 0)
                /* check error */
                ...
        else
                /* can now determine column structure */
                ...
        ...
}
```

```
TEXSESSION      session;
int             contact;
char            cmd[128];
TEXCURSOR       cursor;
...
contact = 9;
...
sprintf( cmd, "delete from contacts
                        where contactno = %d", contact);
if (TexCommand(session,  cmd, &cursor) < 0)
        /* check  error */
        ...
else
        /* deletion was successful */
```

SEE ALSO

TexError, TexErrMsg, TexErrOff, TexColCursor, TexClose

Cursor Type

NAME

TexType - type of cursor

SYNOPSIS

```
int
TexType(cursor, type)
TEXCURSOR        cursor;
int              *type;
```

DESCRIPTION

Determines the type of a cursor. The type is assigned to the type variable passed as a parameter.

Cursor types are: TEXCURQUERY
TEXCURDESCRIBE
TEXCURINSERT
TEXCURUPDATE
TEXCURDELETE

RETURN VALUES

0	Cursor type determined successfully.
-1	Unable to determine cursor type.

ERRORS

TECURBAD Bad cursor.

EXAMPLE

```
TEXCURSOR        curs;
int              curtype;
...
if (TexType(curs, &curtype) < 0)
        /* check error */
        ...
...
switch (curtype)
{
    case  TEXCURQUERY:
        printf("cursor from Texql  query\n");
        break;
    case  TEXCURDELETE:
        printf("cursor from Texql  delete\n");
        break;
```

SEE ALSO

TexCommand

Close Cursor

NAME

TexClose - close a cursor

SYNOPSIS

```
int
TexClose(cursor)
TEXCURSOR       cursor;
```

DESCRIPTION

Closes and frees a cursor. The cursor may be an outer cursor or a nested cursor. Closing a cursor will result in closure of all associated nested cursors.

RETURN VALUES

0	The cursor was closed successfully.
-1	The cursor could not be closed.

ERRORS

TECURBAD Bad cursor.

EXAMPLE

```
TEXCURSOR       cursor;
...
if (TexCommand("select...", &cursor) < 0)
...
TexClose(cursor);
```

SEE ALSO

TexCommand,TexCursor

Merge Cursors

NAME

TexMerge - merge two cursors

SYNOPSIS

```
int
TexMerge(cursor1, cursor2,  dups)
TEXCURSOR       cursor1, cursor2;
TEXS32          *dups;
```

DESCRIPTION

Merge the two cursors, cursor1 and cursor2. More specifically the rows in cursor2 are appended to cursor1, with duplicates being removed. The cursor2 is left unchanged. The number of duplicates removed is set in dups.

Both cursors must be referencing the same base table.

RETURN VALUES

0	Cursors were merged successfully.
-1	Cursors could not be merged.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEBASETABLE	Not a base table cursor.
TEMRGNAMESBAD	Cursors reference different base tables.

EXAMPLE

```
TEXCURSOR       cursor1, cursor2;
TEXS32          dups;
...
if (TexCommand("contacts where surname = 'Johnson',
                                  &cursor1, & dups) < 0)
        ...
if (TexCommand("contacts where postcode = '3220',
                                  &cursor2, & dups) < 0)
        ...
if (TexMerge(cursor1, cursor2, & dups) < 0)
        ...
printf("%d duplicates",  dups);
```

SEE ALSO

TexCommand,TexCursor

Sort Cursor

NAME

TexSort - sort a cursor

SYNOPSIS

```
int
TexSort(cursor,  sortinfo)
TEXCURSOR        cursor;
TEXSORT          sortinfo;
```

DESCRIPTION

Sort the rows of the cursor. The argument sortinfo contains an array of column names and sorting direction flags. The array must be terminated by an element with a NULL column name.

This function resets the row marker to the first row.

RETURN VALUES

0	Sorting was performed successfully.
-1	Sorting failed.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TECOLNAMEBAD	Bad column name.
TESORTCURSOR	Sorting error.

EXAMPLE

```
TEXCURSOR        cursor;
TEXSORT          sortinfo[3];
...
sortinfo[0]. s_colname = "surname";
sortinfo[0]. s_flags =  SORT_ASCEND;
sortinfo[1]. s_colname = " firstnam";
sortinfo[1]. s_flags =  SORT_DESCEND;
sortinfo[2]. s_colname = ( TEXSTRING) NULL;
sortinfo[2]. s_flags = 0;  /* doesn't really matter */
if (TexSort(cursor,  sortinfo) < 0)
        /* error */
```

SEE ALSO

TexCommand,TexCursor

Chapter 5

Row Access

Overview

The TexRow() group of commands can be used to manipulate the row marker which indicates the next row to be accessed.

Next Row

NAME

TexRowNext - retrieve next row

SYNOPSIS

```
int
TexRowNext(cursor)
TEXCURSOR       cursor;
```

DESCRIPTION

Retrieves the next row of the table associated with the cursor. The data is loaded into internal buffers in readiness for access using the TexColDataGet() function.

Repeated calls to TexRowNext() will eventually result in the TEEOF error code being set when all rows are exhausted. A TEEOF error code on a TexRowNext() call should not be viewed as an error but rather an indication that there is no more row data for the specified cursor.

A call to theTexRowGet() function alters the current row marker.

RETURN VALUES

0	The next row was retrieved successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEEOF	No more rows.

EXAMPLES

```
TEXCURSOR       cursor;
TEXSTRING       name;
...
if (TexCommand("contacts where  contno = 13", &cursor) < 0)
...
if (TexRowNext(cursor) < 0)
        /* check error */
...
if (TexColDataGet(cursor, "surname", &name) < 0)
        /* check error */
...
```

```
printf("Surname of contact 13 is %  s\n", name);
TEXCURSOR       cursor;
TEXSTRING       name;
...
printf("Loan  types\n");
if ( TexCommand(" loantypes", &cursor) < 0)
        /* check error */
while ( TexRowNext(cursor) == 0)
{
        if ( TexColDataGet(cursor, " loanname", &name) < 0)
                /* check error */
        printf("% s\n", name);
}
if ( TexError() !=  TEEOF)
        /* real error */
...

TEXCURSOR       cursor, catcur;
TEXSTRING       category;
...
if ( TexCommand("loans where  contno = 13",&cursor) < 0)
...
if ( TexRowNext(cursor) < 0)
...
if ( TexColCursor(cursor, " category_tab", & catcur) < 0)
...
printf("Loan categories of contact 13:\n");
while( TexRowNext( catcur) == 0)
{
        ...
        if ( TexColDataGet( catcur, " category",&category) <
0)
                /* check error */
        ...
        printf("% s\n", category);
        ...
```

SEE ALSO

TexRowGet, TexColDataGet

Get Row

NAME

TexRowGet - retrieve a specific row number

SYNOPSIS

```
int
TexRowGet(cursor,  rownum)
TEXCURSOR        cursor;
TEXS32           rownum;
```

DESCRIPTION

Retrieves the specified row of the table associated with the cursor. The data is loaded into internal buffers in readiness for access using the TexColDataGet() function. Rows are numbered from 1 to TexRowCnt(cursor). Subsequent calls to TexRowNext() will retrieve rows commencing from one greater than the row specified in the TexRowGet() call.

RETURN VALUES

0	The next row was retrieved successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEEOF	The rownum is out of range.

EXAMPLE

```
TEXCURSOR        cursor;
TEXS32           numrows, row;

if (TexCommand("order contacts on exposure", &cursor) < 0)
        /* check error */
if (TexRowCnt(cursor, & numrows) < 0)
        /* check error */
/* print rows in reverse order */
for (row =  numrows; row; row--)
{
        if (TexRowGet(cursor, row) < 0)
                /* check error */
        /* print row */
        ...
```

SEE ALSO

TexRowNext, TexColDataGet

Move Row

NAME

TexRowMove - move the row number relative to the current position

SYNOPSIS

```
int
TexRowMove(cursor,  rownum)
TEXCURSOR        cursor;
TEXS32           rownum;
```

DESCRIPTION

Retrieves a row of the table associated with the cursor relative to the current row position. The data is loaded into internal buffers in readiness for access using the TexColDataGet() function. The supplied row number may be positive (specifying a move forward) or negative (specifying a move backward). Subsequent calls to TexRowNext() will retrieve rows commencing from one greater than the row specified by the TexRowMove() call.

RETURN VALUES

0	The next row was retrieved successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEEOF	The derivedrownum is out of range.

EXAMPLE

```
TEXCURSOR        cursor;
TEXS32           numrows, row;

if (TexCommand("contacts", &cursor) < 0)
        /* check error */
if (TexRowCnt(cursor, & numrows) < 0)
        /* check error */
/* print rows in reverse order */
for (row =  numrows; row; row--)
{
        ...
        if (TexRowMove(cursor, -1) < 0)
                /* check error */
```

SEE ALSO

TexRowGet, TexColDataGet

Row Position

NAME

TexRowPos - determine row marker position

SYNOPSIS

```
int
TexRowPos(cursor,  rownum)
TEXCURSOR       cursor;
TEXS32          *rownum;
```

DESCRIPTION

Determine the current row number position of the row marker.

RETURN VALUES

0	Row marker determined.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.

EXAMPLE

```
TEXCURSOR       cursor;
TEXS32          rownum;
...
if (TexRowPos(cursor, & rownum) < 0)
        /* check error */
...
printf("Row marker at row % ld\n", (long)  rownum);
```

SEE ALSO

TexRowNext, TexRowGet, TexRowReset

Row Reset

NAME

TexRowReset - reset the cursor row marker

SYNOPSIS

```
int
TexRowReset(cursor)
TEXCURSOR       cursor;
```

DESCRIPTION

Resets the row marker associated with the cursor such that the next call to TexRowNext() will retrieve the first row of the table.

RETURN VALUES

0	Reset successfully completed.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.

EXAMPLE

```
TEXCURSOR       cursor;
...
if (TexRowGet(cursor, (TEXS32) 7) < 0)
        /* check error */
...
/* process row 7 */
...
if (TexRowReset(cursor) < 0)
        /* check error */
...
/* start processing again from row 1 */
while (TexRowNext(cursor) == 0)
{
        ...
```

SEE ALSO

TexRowNext

Count Rows

NAME

TexRowCnt - count the number of rows

SYNOPSIS

```
int
TexRowCnt(cursor,  nrows)
TEXCURSOR       cursor;
TEXS32          *nrows;
```

DESCRIPTION

Counts the number of rows associated with the cursor. For a Texql query command, TEXCURQUERY, this function will assign to the parameter the number of rows that matched the query.

For a Texql data manipulation command, TEXCURINSERT, TEXCURUPDATE or TEXCURDELETE, this function will assign to the parameter the number of rows that were inserted, updated or deleted respectively.

This function resets the row marker to the first row of the table.

RETURN VALUES

0	Row count successfully accessed.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECOLTYPE	Bad column type.

EXAMPLE

```
TEXCURSOR       cursor;
TEXS32          numloans;

...
if (TexCommand("loans", &cursor) < 0)
...
if (TexRowCnt(cursor, & numloans) < 0)
        /* check error */
...
printf("Table contains % ld loans\n", (long)  numloans);
...
```

SEE ALSO

TexRowNext, TexRowGet, TexRowReset

Number of Row Hits

NAME

TexRowHits - determine the number of row hits

SYNOPSIS

```
int
TexRowHits(cursor,  nhits)
TEXCURSOR        cursor;
TEXS32           *nhits;
```

DESCRIPTION

Determines the number of hits associated with a query cursor (TEXCURQUERY. For query commands in which column attributes were provided that enabled the index to be utilised this function sets the number of row hits. In this case TexRowHits is significantly faster than TexRowCnt.

If indexing information was not able to be utilised by the query then -1 is returned in the hits variable and TexRowCnt must be used to determine the number of matching rows.

This function resets the row marker to the first row of the table.

RETURN VALUES

0	Function completed normally.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.

EXAMPLE

```
TEXCURSOR        cursor;
TEXS32           hits;

...
if (TexCommand("loans where term = 12", &cursor) < 0)
...
if (TexRowHits(cursor, &hits) < 0)
        /* check error */
if (hits == (TEXS32) -1)
        /* unable to determine hits */
else
        printf("Hit % ld rows\n", (long) hits);
```

SEE ALSO

TexRowNext, TexRowGet, TexRowCnt, TexRowReset

Lock Row

NAME

TexRowLock - lock a row

SYNOPSIS

```
int
TexRowLock(cursor,  rownum)
TEXCURSOR        cursor;
TEXS32           rownum;
```

DESCRIPTION

Place a lock on the rownum'th row of the cursor. This provides the cursor with exclusive update access to that row. Locking will fail if some other cursor or Texpress program has already obtained an exclusive row lock.

If the rownum is TEXROWCURRENT then the current row is locked. If the rownum isTEXROWALL then all rows of the cursor are locked.

The cursor must be referencing a base table.

RETURN VALUES

0	The row(s) was locked successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEROWLOCK	Lock failed.

EXAMPLE

```
TEXCURSOR        cursor;

...
/* lock row 3 */
if (TexRowLock(cursor, (TEXS32) 3) < 0)
        /* check error */
...
```

SEE ALSO

TexRowUnlock,TexRowStatus

Unlock Row

NAME

TexRowUnlock - unlock a row

SYNOPSIS

```
int
TexRowUnlock(cursor,  rownum)
TEXCURSOR        cursor;
TEXS32           rownum;
```

DESCRIPTION

Unlock on the rownum'th row of the cursor. It is only possible to unlock a row that was previously locked by a call to TexRowLock using the same cursor.

If the rownum is TEXROWCURRENT then the current row is unlocked. If the rownum is TEXROWALL then all rows of the cursor are unlocked.

The cursor must be referencing a base table.

RETURN VALUES

0	The row was unlocked successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEROWUNLOCK	Unlock failed.

EXAMPLE

```
TEXCURSOR        cursor;

...
/* unlock row 3 */
if (TexRowUnlock(cursor, (TEXS32) 3) < 0)
        /* check error */
...
```

SEE ALSO

TexRowLock, TexRowStatus

Row Status

NAME

TexRowStatus - determine status of a row

SYNOPSIS

```
int
TexRowStatus(cursor,  rownum, status)
TEXCURSOR        cursor;
TEXS32           rownum;
TEXU32           *status;
```

DESCRIPTION

Determine the status of the rownum'th row of the cursor. The status indicates if the row has been locked. If the row is the current row the status also indicates if the row has been modified. The cursor must be referencing a base table.

The status flag is a bit map where any of the following may be set:

TEXROWMODIFIED	/* row has been modified */
TEXROWCURSORLOCK	/* row locked by this cursor */
TEXROWOTHERLOCK	/* row locked by other cursor */
TEXROWUPDATED	/* row updated by other cursor */
TEXROWDELETED	/* row deleted by other cursor */

RETURN VALUES

0	Row status determined successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TEROWSTATUS	Row status failed.

EXAMPLE

```
TEXCURSOR        cursor;
TEXU32           status;

if (TexRowStatus(cursor, (TEXS32) 3, &status) < 0)
        /* check error */
if (status &  TEXROWOTHERLOCK)
        ...
```

SEE ALSO

TexRowLock, TexRowUnlock

New Row

NAME

TexRowNew - create a new row

SYNOPSIS

```
int
TexRowNew(cursor,  rownum,  keydata)
TEXCURSOR        cursor;
TEXS32           rownum;
TEXARRAY         *keydata;
```

DESCRIPTION

Create a new row in the cursor. The row is created imediately before the row at position, rownum, or if rownum is TEXROWAPPEND or is greater than TexRowCnt() the row is added to the end of the table. The row marker is set to the new row. The cursor must be referencing a base table.

The keydata parameter is used to specify the primary key value to be assigned to the new row. It comprises an NULL terminated array of TEXSTRING values which are assigned in turn to each of the columns of the primary key tuple. If the table does not have a primary key then the primary key parameter must be NULL.

If keydata is NULL or values are provided for only some of the primary key columns then, if possible, an automatic primary key will be assigned.

A new row does not become permanent until TexRowSave() is called. A new row mau be discarded prior to saving by calling TexRowDiscard() with the appropriate row number.

Following a TexRowNew() call and prior to any TexRowSave() or TexRowDiscard() call, calls made to any other TexRow function which may move the row marker will result in the new row being silently discarded.

RETURN VALUES

0	Row created successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEKEYNONE	Table does not have a primary key.
TEKEYFAIL	Failed to assign primary key.
TEKEYBAD	Badly formed primary key.
TEKEYDUP	Duplicate primary key.

EXAMPLE

```
TEXCURSOR       cursor;
TEXSTRING       keydata[2];

...
/* Create a new row at row position 3 of the cursor,

** and assign a primary key value of 10.
*/
keydata[0] = "10";
keydata[1] = ( TEXSTRING) NULL;
if (TexRowNew(cursor, (TEXS32) 3,  keydata) < 0)
        /* check error */
...
/*  perform further editing using TexColDataSet()
*/
...
if (want to permanently save row)
{
        if (TexRowSave(cursor) < 0)
                /* check error */
}
else    /* discard row */
{
        if (TexRowDiscard(cursor,  TEXROWCURRENT) < 0)
                /* check error */
}
```

SEE ALSO

TexRowSave, TexRowDiscard

Save Row

NAME

TexRowSave - save the current row

SYNOPSIS

```
int
TexRowSave(cursor)
TEXCURSOR       cursor;
```

DESCRIPTION

Save the current row of the cursor. This creates or updates a permanent row in the table. The cursor must be referencing a base table.

The TEXROWMODIFIED flag is cleared by this function. A row lock is not removed by this function.

RETURN VALUES

0	Row status determined successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TEROWSTATUS	Row status failed.

EXAMPLE

```
TEXCURSOR       cursor;

...
if (TexRowSave(cursor) < 0)
        /* check error */
```

SEE ALSO

TexRowNew, TexColDataSet, TexRowDiscard

Discard Row

NAME

TexRowDiscard - discard a row

SYNOPSIS

```
int
TexRowDiscard(cursor,  rownum)
TEXCURSOR       cursor;
TEXS32          rownum;
```

DESCRIPTION

Discard the rownum'th row of the cursor. If rownum is TEXROWCURRENT then the current row is discarded. This discards the row from the current cursor only, it does not delete the row from the table.

The cursor must be referencing a base table.

After a call to TexRowNew(), but prior to a call to TexRowSave(), TexRowDiscard() may be called to discard the new row.

RETURN VALUES

0	Row discarded successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.

EXAMPLE

```
TEXCURSOR       cursor;

...
/* discard row 3 from cursor */
if (TexRowDiscard(cursor, (TEXS32) 3) < 0)
        /* check error */
```

SEE ALSO

TexRowDelete,TexRowNew

Delete Row

NAME

TexRowDelete - delete a row

SYNOPSIS

```
int
TexRowDelete(cursor,  rownum)
TEXCURSOR        cursor;
TEXS32           rownum;
```

DESCRIPTION

Delete the rownum'th row of the cursor. If rownum is TEXROWCURRENT then the current row is deleted. This permanently deletes the row from the table.

The cursor must be referencing a base table.

RETURN VALUES

0	Row deleted successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNOTQRY	Not a query cursor.
TECURSDESC	This is a describe cursor.
TELOCKREC	Unable to lock row.

EXAMPLE

```
TEXCURSOR        cursor;

...
/* delete row 3 */
if (TexRowDelete(cursor, (TEXS32) 3) < 0)
        /* check error */
```

SEE ALSO

TexRowDiscard

Chapter 6

Column Access

Overview

The TexCol() group of functions can be used to access the column structure for the operation as well as get and set column data values.

Column Names

NAME

TexColNames - access column names

SYNOPSIS

```
int
TexColNames(cursor,  colnames)
TEXCURSOR        cursor;
TEXARRAY         *colnames;
```

DESCRIPTION

Retrieves an ordered list of the column names associated with the cursor and assigns it to the colnames parameter. Upon successful completion colnames will point to a NULL terminated array of column names.

RETURN VALUES

0	Column names accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECOLTYPEBAD	Bad column type.

EXAMPLE

```
TEXCURSOR        cursor;
TEXARRAY         loannames;
int              i;
...
if (TexCommand("describe loans", &cursor) < 0)
...
if (TexColNames(cursor, & loannames) < 0)
        /* check error */
...
for (i = 0;  loannames[ i]; i++)
        printf("%s\n",  loannames[ i]);
```

SEE ALSO

TexColKind,TexColType

Column Kind

NAME

TexColKind - determine column kind

SYNOPSIS

```
int
TexColKind(cursor,  colname, kind)
TEXCURSOR        cursor;
TEXSTRING        colname;
int              *kind;
```

DESCRIPTION

Retrieves the kind of column for colname. The colname must be the name of a valid column of the table or tuple associated with cursor.

Columns kinds are: TEXKINDTABLE
TEXKINDTUPLE
TEXKINDATOM

RETURN VALUES

0	Column kind accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECOLNAMEBAD	Bad column name.
TECOLTYPEBAD	Bad column type.

EXAMPLE

```
TEXCURSOR        cursor;
int              kind;
...
if (TexCommand("describe  loantypes", &cursor) < 0)
...
if (TexColKind(cursor, " modon", &kind) < 0)
        /* check error */
switch (kind)
{
    ...
    case  TEXKINDTUPLE:
         printf("modon is a  tuple\n");
         break;
         ...
```

SEE ALSO

TexColType

Column Type

NAME

TexColType - determine column type

SYNOPSIS

```
int
TexColType(cursor,  colname, type)
TEXCURSOR       cursor;
TEXSTRING       colname;
int             *type;
```

DESCRIPTION

Retrieves the type of column for the atomic column colname. The colname must be a valid atomic column of the table or tuple associated with cursor.

Columns types are: TEXTYPETEXT
TEXTYPEINTEGER
TEXTYPEFLOAT

RETURN VALUES

0	Column kind accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECOLNAMEBAD	Bad column name.
TECOLTYPEBAD	Not an atomic column type.
TEATOMTYPEBAD	Bad atom type.

EXAMPLE

```
TEXCURSOR       cursor;
int             type;
...
if (TexCommand("loans where  loanno = 10", &cursor) < 0)
...
if (TexColType(cursor, "amount", &type) < 0)
        /* check error */
...
if (type !=  TEXTYPEFLOAT)
        printf("amount must be a real  value\n");
...
```

SEE ALSO

TexColKind

Column Nested Cursor

NAME

TexColCursor - obtain a nested cursor

SYNOPSIS

```
int
TexColCursor(cursor,  colname, nested)
TEXCURSOR       cursor;
TEXSTRING       colname;
TEXCURSOR       *nested;
```

DESCRIPTION

Obtain a nested cursor for the column name colname associated with cursor. This nested cursor may then used in subsequent TexRow() and TexCol() calls. Typically a nested cursor is used to access values in a nested table or tuple. A nested cursor is closed using TexClose(). Closing a cursor will result in closure of all associated nested cursors.

RETURN VALUES

0	Nested cursor obtained successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECOLNAMEBAD	Bad column name.
TECOLTYPEBAD	Not an atomic column type.

EXAMPLE

```
TEXCURSOR       cursor,  datecur;
int             type;

if (TexCommand("describe  loantypes", &cursor) < 0)
...
if (TexColCursor(cursor, " modon", &datecur) < 0)
        /* check error */
...
if (TexColType( datecur, "modon_2", &type) < 0)
        /* check error */
switch (type)
{
    case  TEXTYPEINTEGER:
        printf("modon: second field is  integer\n");
        break;
```

SEE ALSO

TexColNames,TexColKind

Column Data Get

NAME

TexColDataGet - access data for an atomic column

SYNOPSIS

```
int
TexColDataGet(cursor,  colname, data)
TEXCURSOR       cursor;
TEXSTRING       colname;
TEXSTRING       *data;
```

DESCRIPTION

Access the data of column colname associated with the cursor. The column must be an atomic column.

If the data value is NULL then the data variable is assigned a NULL pointer.

RETURN VALUES

0	Data obtained successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECOLNAMEBAD	Bad column name.
TECOLTYPEBAD	Not an atomic column type.

EXAMPLES

```
TEXCURSOR       cursor;
TEXSTRING       name;
...
if (TexCommand("contacts where  contno = 13", &cursor) < 0)
...
if (TexRowNext(cursor) < 0)
        /* check error */
...
if (TexColDataGet(cursor, "surname", &name) < 0)
        /* check error */
...
printf("Surname of contact 13 is %  s\n", name);
```

```
TEXCURSOR       cursor;
TEXSTRING       name;
...
printf("Loan  types\n");
if (TexCommand(" loantypes", &cursor) < 0)
        /* check error */
while ( TexRowNext(cursor) == 0)
{
        if (TexColDataGet(cursor, " loanname",&name) < 0)
                /* check error */
        printf("%s\n", name);
}
if (TexError() !=  TEEOF)
        /* real error */
...

TEXCURSOR       cursor, catcur;
TEXSTRING       category;
...
if (TexCommand("loans where  contno = 13",&cursor) < 0)
...
if (TexRowNext(cursor) < 0)
...
if (TexColCursor(cursor, " category_tab", & catcur) < 0)
...
printf("Loan categories of contact 13:\n");
while( TexRowNext( catcur) == 0)
{
        ...
        if (TexColDataGet( catcur, "category",&category) <
0)
                /* check error */
        ...
        printf("%s\n", category);
        ...
```

SEE ALSO

TexColKind,TexColType,TexColCursor,TexColDataSet

Column Data Set

NAME

TexColDataSet - assign data for an atomic column

SYNOPSIS

```
int
TexColDataSet(cursor,  colname, data)
TEXCURSOR       cursor;
TEXSTRING       colname;
TEXSTRING       data;
```

DESCRIPTION

Set the column, colname, to the value, data. Updates using the function may only be made to columns of base tables.

This command sets the in memory value only. Data is not permanently stored until TexRowSave() is called.

The TEXROWMODIFIED row flag is set by this function.

RETURN VALUES

0	Data assigned successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECOLNAMEBAD	Bad column name.
TECOLTYPEBAD	Not an atomic column type.

EXAMPLES

```
TEXCURSOR       cursor;
TEXSTRING       name;

...
if (TexCommand("contacts where  contno = 13", &cursor) < 0)
...
if (TexRowNext(cursor) < 0)
        /* check error */
...
if (TexColDataSet(cursor, "surname", "  Roberts") < 0)
        /* check error */
```

SEE ALSO

TexColCursor, TexColDataGet, TexRowStatus, TexRowSave

Chapter 7

Convenience Functions

Overview

KE Texpress Texql and the C-API have inherent support for data structures more flexible than those provided by version 3.4 databases. To assist programmers in utilising the C-API several version 3.4 convenience functions are provided. These convenience functions provide simplified access to existing version 3.4 databases.

Item Names

NAME

TexItmNames - access item names (3.4 convenience function)

SYNOPSIS

```
int
TexItmNames(cursor,  itmnames)
TEXCURSOR        cursor;
TEXARRAY         *itmnames;
```

DESCRIPTION

Retrieves an ordered list of the item (column) names associated with the cursor and assigns it to the itmnames parameter. Upon successful completion itmnames will point to a NULL terminated array of item names.

The cursor must be accessing an outer table.

RETURN VALUES

0	Item names accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNEST	Not available for nested cursors.

EXAMPLE

```
TEXCURSOR        cursor;
TEXARRAY         loannames;
int              i;
...
if (TexCommand("describe loans", &cursor) < 0)
...
if (TexItmNames(cursor, & loannames) < 0)
        /* check error */
...
for (i = 0;  loannames[ i]; i++)
        printf("%s\n",  loannames[ i]);
...
```

SEE ALSO

TexColNames,TexItmFlds,TexItmDataGet,TexItmDataSet

Item Number of Fields

NAME

TexItmFlds - number of fields of an item (3.4 convenience function)

SYNOPSIS

```
int
TexItmFlds(cursor,  itmname, count)
TEXCURSOR        cursor;
TEXSTRING        itmname;
int              *count;
```

DESCRIPTION

Obtains the number of fields of itmname and assigns it to the count variable. The cursor must be accessing an outer table.

RETURN VALUES

0	Accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNEST	Not available for nested cursors.
TECOLNAMEBAD	Bad item name.

EXAMPLE

```
TEXCURSOR        cursor;
int              nflds;
...
if (TexCommand("describe contacts", &cursor) < 0)
...
if (TexItmFlds(cursor, " maillist", & nflds) < 0)
        /* check error */
...
printf("Mailing list has %d  fields\n",  nflds);
```

SEE ALSO

TexItmNames,TexItmDataGet

Item Data Get

NAME

TexItmDataGet - access item names (3.4 convenience function)

SYNOPSIS

```
int
TexItmDataGet(cursor,  itmname, data)
TEXCURSOR       cursor;
TEXSTRING       itmname;
TEXARRAY        *data;
```

DESCRIPTION

Accesss all the data associated with item itmname and assigns it to the variable data. Upon successful completion data will contain TexItmFlds data value pointers.

The cursor must be accessing an outer table.

RETURN VALUES

0	Item data accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNEST	Not available for nested cursors.
TECOLNAMEBAD	Bad item name.

EXAMPLES

```
TEXCURSOR       cursor;
int             nflds;
TEXARRAY        data;
int             i;

..
if (TexCommand("contacts where  contno = 42",
                                  &cursor) < 0)
...
if (TexItmFlds(cursor, " maillist", & nflds) < 0)
        /* check error */
if (TexItmDataGet(cursor, " maillist", &data) < 0)
        /* check error */
for (i = 0;  i < nflds && data[ i]; i++)
        printf("%d: % s\n",  i, data[ i]);
...
```

```
TEXCURSOR        curs;
TEXARRAY         modon;
...
if (TexCommand(" loantypes where  loanname contains
'travel'", &curs) < 0)
...
if (TexItmDataGet(curs, " modon", &modon) < 0)
        /* check error */
...
printf("Date is %s/%s/ %s\n",
                          modon[0],  modon[1],  modon[2]);
...
```

SEE ALSO

TexItmNames,TexItmFlds,TexItmDataSet

Item Data Set

NAME

TexItmDataSet - access item names (3.4 convenience function)

SYNOPSIS

```
int
TexItmDataSet(cursor,  itmname, data)
TEXCURSOR       cursor;
TEXSTRING       itmname;
TEXARRAY        data;
```

DESCRIPTION

Set the fields of the item, itmname, to the array of text strings, data. The cursor must be accessing a base table.

This command sets the in memory value only. Data is not permanently stored until TexRowSave() is called.

The TEXROWMODIFIED row flag is set by this function.

RETURN VALUES

0	Item data assigned successfully .
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNEST	Not available for nested cursors.
TECOLNAMEBAD	Bad item name.

EXAMPLES

```
TEXCURSOR       cursor;
TEXSTRING       *data[4];

..
if (TexCommand("contacts", &cursor) < 0)
...
data[0] = "Travel";
data[1] = "Home improvement";
data[2] = "Boating";
data[3] = ( TEXSTRING) NULL;
if (TexItmDataSet(cursor, " maillist", data) < 0)
        /* check error */
```

SEE ALSO

TexItmNames, TexItmFlds, TexItmDataGet, TexRowSave

Field Type

NAME

TexFldType - access the field type (3.4 convenience function)

SYNOPSIS

```
int
TexFldType(cursor,  itmname,  fldno, type)
TEXCURSOR       cursor;
TEXSTRING       itmname;
int             fldno;
int             *type;
```

DESCRIPTION

Accesss the type of field fldno for item itmname and assign it to the type variable. The fldno must be in the range 1 to TexItmFlds(). The cursor must be accessing an outer table.

RETURN VALUES

0	Field type accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNEST	Not available for nested cursors.
TECOLNAMEBAD	Bad item name.
TEFIELDBAD	Bad field number.

EXAMPLE

```
TEXCURSOR       cursor;
int             type;
...
if (TexCommand("describe  loantypes", &cursor) < 0)
...
if (TexFldType( datecur, " modon", 2, &type) < 0)
        /* check error */
...
switch (type)
{
    ...
    case  TEXTYPEINTEGER:
        printf(" modon: second field is  integer\n");
        break;
    ...
```

SEE ALSO

TexItmNames, TexItmFlds, TexFldType

Field Data Get

NAME

TexFldDataGet - access field data (3.4 convenience function)

SYNOPSIS

```
int
TexFldDataGet(cursor,  itmname,  fldno, data)
TEXCURSOR       cursor;
TEXSTRING       itmname;
int             fldno;
TEXSTRING       *data;
```

DESCRIPTION

Accesss the data of field number fldno for item itmname and assigns it to the data variable. Thefldno must be in the range 1 toTexItmFlds().

The cursor must be accessing an outer table.

RETURN VALUES

0	Item data accessed successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNEST	Not available for nestedcursors.
TECOLNAMEBAD	Bad item name.
TEFIELDBAD	Bad field number.

EXAMPLE

```
TEXCURSOR       curs;
char            *month;
...
if (TexCommand(" loantypes where  loanname
                        contains 'travel'", &curs) < 0)
...
if (TexFldDataGet(curs, " modon", 2, &month) < 0)
        /* check error */
...
printf("Month is % s\n", month);
...
```

SEE ALSO

TexItmNames,TexItmFlds,TexFldType,TexFldDataSet

Field Data Set

NAME

TexFldDataSet - assign field data (3.4 convenience function)

SYNOPSIS

```
int
TexFldDataSet(cursor,  itmname,  fldno, data)
TEXCURSOR        cursor;
TEXSTRING        itmname;
int              fldno;
TEXSTRING        data;
```

DESCRIPTION

Assign the data to field number fldno of item itmname. The fldno must be in the range 1 to TexItmFlds(). The cursor must be accessing an outer table.

This command sets the in memory value only. Data is not permanently stored until TexRowSave() is called.

The TEXROWMODIFIED row flag is set by this function.

RETURN VALUES

0	Data assigned successfully.
-1	An error occurred.

ERRORS

TECURBAD	Bad cursor.
TECURSNEST	Not available for nested cursors.
TECOLNAMEBAD	Bad item name.
TEFIELDBAD	Bad field number.

EXAMPLE

```
TEXCURSOR        curs;

...
if (TexCommand(" loantypes where  loanname
                          contains 'travel'", &curs) < 0)
...
if (TexFldDataSet(curs, " modon", 2, "11") < 0)
        /* check error */
```

SEE ALSO

TexItmNames,TexItmFlds,TexFldDataGet,TexRowSave

Appendix A

Sample Program

This example program accesses the contacts table and outputs the contact name, position and mailing list details for each person. An ordered Texql query is used to sort the output on surname.

```
/*
** A sample KE Texpress C-API program.
** This program prints out the Name, Position, and Mailing List
** of each person in the "contacts" table.
** The information is sorted by surname.
*/
#include        <stdio.h>
#include        "texapi.h"


#define  CMD    "order contacts[ firstnam, surname, position, \
                maillist_tab] on surname"


void            print           _FP((TEXCURSOR));
void            error           _FP((char *));


void
main(argc, argv)
int     argc;
char    **argv;
{
        TEXPARAMS       params;
        TEXSESSION      session;
        TEXCURSOR       cursor;

        if (TexInitialise(&argc, argv, &params) < 0)
                error("TexInitialise");
        if (TexConnect(&params, &session) < 0)
                error("TexConnect");
        if (TexCommand(session,  CMD, &cursor) < 0)
                error("TexCommand");
        print(cursor);
        TexClose(cursor);
        TexDisconnect(session);
        TexTerminate();
        exit(0);
}
```

```
void
print(cursor)
TEXCURSOR       cursor;
{
        TEXCURSOR       mailcur;
        TEXSTRING       first, surname,  pos, mail;
        char            name[50];

        if (TexColCursor(cursor, " maillist_tab", & mailcur) < 0)
                error("TexColCursor");
        printf("Name          Position          Mailing   List\n");
        printf("----          --------          ------------\n");
        while (TexRowNext(cursor) == 0)
        {
                if (TexColDataGet(cursor, " firstnam", &first) < 0)
                        error("TexColDataGet - first");
                if (TexColDataGet(cursor, "surname", &surname) < 0)
                        error("TexColDataGet - surname");
                if (surname == (char *) NULL)
                        name[0] = '\0';
                else if (first == (char  *) NULL)
                        strcpy(name, surname);
                else
                        sprintf(name, "%s, %s", surname, first);
                if (TexColDataGet(cursor, "position", &  pos) < 0)
                        error("TexColDataGet - position");
                if (pos == (char *) NULL)
                        pos = "";
                while (TexRowNext( mailcur) == 0)
                {
                        if (TexColDataGet( mailcur,"maillist", &mail) < 0)
                                error("TexColDataGet - mail");
                        printf("%-20s %-18s % s\n", name,  pos, mail);
                        name[0] = '\0';
                        pos = "";
                }
                if (TexError() !=  TEEOF)
                        error("TexRowNext -  mailcur");
        }
        if (TexError() !=  TEEOF)
                error("TexRowNext - cursor");
        TexClose( mailcur);
}


void
error( msg)
char    *msg;
{
        char    *str;

        fprintf( stderr, "API call failed: %s: error no = %  d\n",   msg,
TexError());
        if ((str =  TexErrMsg()) &&  str[0])
                fprintf( sdterr, "Message: % s\n",  str);
        TexTerminate();
        exit(1);
}
```

Appendix B

Error Codes

001 "Internal error: %s"
002 "Expression failed"
003 "Link to REF failed"
004 "Validation failed"
005 "Permission denied"
006 "Table is readonly"
007 "Can't find \"%s\" table"
008 "You are not a registered user of \"%s\" table"
009 "Database \"%s\" not initialised"
010
011 "Database startup failed"
012 "Failed to read table"
013
014
015 "Cannot lock data file"
016 "Cannot lock duplicate data file"
017 "Cursor is not a query cursor"
018 "End of file"
019 "No more cursors available"
020 "Bad cursor"
021 "Can't determine user identity"
022 "Query does not return an atomic value"
023 "Column \"%s\" is of incorrect type "
024 "Unknown column name \"%s\""
025 "Atomic column %s has unknown (bad) type"
026 "Describe cursor cannot access data"
027 "Operation is not permitted on a nested cursor"
028 "Cursor is not a reference to KE Texpress database"
029 "Bad item name"
030 "Bad field number"
031 "Column operation performed before row has been accessed"
032 "Nested cursor operation performed before row has been accessed"
033 "The API cannot be run by the superuser"
034 "Permission denied"
035 "Operation interrupted by front-end"
036 "Column \"%s\" not a base KE Texpress table"
037 "Row lock failed"

038 "Row unlock failed"
039 "Row status failed"
040 "Merge arguments refer to different base tables"
041 "Merge arguments table paths differ"
042 "Column \"%s\" is read only"
043 "Sort of cursor failed"
044 "Incompatible versions of client library and KE Texpress server"
045 "Function not yet implemented in KE Texpress server"
046 "Reference column not permitted"
047 "New row has not been saved or discarded"
048 "Table does not have a primary key"
049 "Failed to assign primary key"
050 "Badly formed primary key"
051 "Duplicate primary key"
052 "Licence error"

200 "BUT caused all columns to be removed"
201 "Unable to resolve BUT identifier"
202 "Can't evaluate expression to atomic value"
203 "Illegal NULL value in expression evaluation"
204 "INSERT BEFORE/AFTER not permitted on base table"
205 "TOTUPLE cannot return row from empty table"
206 "TOTUPLE can only return a row from a table that has only one row"
207 "Too many tables to join on"
208 "Unknown table in PRESERVE clause"
209 ".ident can only be applied to a tuple"
210 ".%s not a column of the tuple"
211 "Unable to resolve \"%s\""
212 "Attribute specification too complex for GROUP"
213 "GROUP operator not being applied to table"
214 "Tuple projection not from tuple expression"
215 "UPDATE only works on tuples or tables"
216 "FROM line not a table expression"
217 "References too complex to follow"
218 "Identifier nesting too deep"
219 "Column number out of range"
220 "Ambiguous identifier"
221 "HAS on non table column"
222 "STEM makes no sense on incomplete word"
223 "PHONETIC makes no sense on incomplete word"
224 "A syntax error has occurred while parsing text"
225 "BUT must come last on SELECT line"
226 "Text constant must have at least one character (otherwise use NULL)"
227 "Incompatible tuples in constant table"
228 "AS expression is type incompatible"

```
229     "Left hand side of '=' must be an identifier"
230     "Identifier \"%s\" recursively defined"
231     "Arithmetic operator can only be applied to atomic types"
232     "Can't use arithmetic operator on type %s"
233     "Can only use IS NULL operator on atomic types"
234     "EXISTS can only be applied to table expressions"
235     "Illegal ROWNUM operator"
236     "Incorrect number of arguments to %s"
237     "%s can only be applied to TEXT values"
238     "COUNT can only be applied to tables"
239     "IFNULL can only be applied to atomic values"
240     "Arguments of IFNULL type incompatible"
241     "Arguments of %s must be atomic"
242     "First argument of %s must be of type TEXT"
243     "Second argument of %s must be of type INTEGER"
244     "%s requires a table that has a single column of atomic values"
245     "DEFAULT value of different type to table column for %s"
246     "%s requires a single numeric column table"
247     "DEFAULT value for %s is not an atomic value"
248     "DEFAULT value must numeric for %s"
249     "%s requires a table expression"
250     "Can only %s on boolean values"
251     "Can't use boolean operator on type %s"
252     "Can only %s on atomic values"
253     "Two sides of %s operator not of compatible types"
254     "Bad left hand side of LIKE"
255     "Right hand side of LIKE must be of type TEXT"
256     "CONTAINS used only on TEXT columns"
257     "Right hand operand of CONTAINS must be a TEXT constant"
258     "Left hand operand of HAS must be a single atomic valued column table"
259     "Right hand operand of HAS must be an atomic value"
260     "%s left and right hand operands are type incompatible"
261     "Left hand operand of IN must be an atomic value"
262     "Right hand operand of IN must be a single atomic valued column table"
263     "%s requires table expressions as operands"
264     "Can only apply BETWEEN to atomic values"
265     "Incompatible types in BETWEEN clause"
266     "WHERE not from a table"
267     "WHERE clause must return a boolean value"
268     "SELECT-FROM-WHERE expression not from a table"
269     "Tuple projection not from a tuple"
270     ".* only permitted on SELECT line"
271     ".* can only be applied to tuples"
272     "%s expression not from a table"
273     "%s must have boolean condition"
```

274 "%s must return a boolean value"
275 "Not a COLUMN to GROUP on"
276 "Illegal GROUP identifier"
277 "Can only GROUP tables"
278 "Can only %s tables"
279 "Can only %s nested tables"
280 "Can only insert into tables"
281 "Tuple structure different to table structure"
282 "Can only modify tables with %s"
283 "Incompatible types for SET"
284 "Range value must be of type INTEGER"
285 "Operand of [] must be of type TABLE"
286 "Referenced table no longer exists"
287 "ORDER cannot be applied to atomic values"
288 "ORDER can only be applied to table values"
289 "Badly formed identifier"
290 "PRESERVE without WHERE clause"
291 "Illegal PRESERVE"
292 "Syntax error"
293 "String not terminated at end of line"
294 "Division by zero"
295 "Modulus by zero"
296
297
298 "Database is full"
299 "Server panic"

Index

A

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V