

RETURN TO MAIN MENU

Recital Environment

Guide to the Recital Environment

Recital Corporation,
100 Cummings Center, Suite 318J
Beverly, MA 01915

Recital may have patents and/or patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.
COPYRIGHT ©1988-2006 Recital Corporation. All rights reserved. All Recital products are trademarks or registered trademarks of Recital Corporation, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Last Updated August, 2006

INDEX

CONSTANTS	1
MEMORY VARIABLES	2
MATHEMATICAL OPERATORS	4
LOGICAL OPERATORS	5
RELATIONAL OPERATORS	6
SPECIAL PURPOSE OPERATORS	7
RECITAL/4GL BASICS	8
PARAMETER PASSING	9
DEFAULT FILE EXTENSIONS	10
COMMANDS INVOKED FROM THE OPERATING SYSTEM	11
RECITAL/LINKER	12
USERS FILES	14
DBEXEC – Running Background Server Programs	15
PRINTING	16
DATABASE TABLE ENCRYPTION	19
FUNCTION KEYS	20
NATIONAL CHARACTER SETS	21
EXTENSIBLE MARKUP LANGUAGE (XML)	23
ERROR HANDLING AND DEBUGGING	24
RECITAL/4GL ERROR MESSAGES	26
SERVER ERROR MESSAGES	33
RECITAL/SQL ERROR MESSAGES	34
ACCESSING RMS FILES	36
ACCESSING C-ISAM FILES	41
C-ISAM RDD ERROR MESSAGES	44
USING XBASE FILES	45
XBASE RDD ERROR MESSAGES	48
ASCII CHART	50
UPGRADE INFORMATION	53
OPTIMIZING INDEXES WITH SYNCNUM	55
ENVIRONMENT VARIABLES / SYMBOLS	
DB_CONFIG	56
DB_DATADIR	57
DB_DATE	58
DB_ENCRYPTION	58
DB_ERRORDIR	59
DB_FIRECATLOG	59
DB_FOXMEM	60
DB_FOXPLUSBUGS	61
DB_FOXPROKEYS	61
DB_HOSTNAME	62
DB_HTTP_ALLOW	63
DB_HTTP_DENY	63
DB_INDEXSEQNO	64
DB_LICENSE_SERVER	65
DB_LOCAL_LOGIN	65
DB_LOGDIR	66
DB_LOGVER	66
DB_MAXROW	67

DB_MAXWKA	67
DB_MIRAGE	68
DB_MIRAGE_COMMAND	69
DB_MIRAGE_CONFIG	69
DB_MIRAGE_DIRECTORY	69
DB_MIRAGE_PATH	70
DB_NOPAM	71
DB_ODBC_INI	71
DB_OPTLOG	72
DB_PRINTEREJECT	73
DB_PRINTERERROR	73
DB_PROCDIR	74
DB_REREAD_COLORFILE	75
DB_RUNLOG	75
DB_RUNOPTS	76
DB_RUNOPTS2	76
DB_SAMBA	77
DB_SAVEHISTORY	77
DB_TMPDIR	78
DB_TSINDEX	79
DB_UAS_ALLOW	80
DB_UAS_DENY	80
DB_UNIXPATH	81
DB_USERLOG	81
DB_XMLREP	82

Constants

The Recital/4GL supports the following constants:

Datatype	Description
Character	A string of ASCII characters up to 8191 characters in length. Delimited by double quotes "", single quotes ' ' or square brackets []
Numeric	An integer number of up to 16 digits (0-9) or a floating point number of up to 25 digits (9 are reserved for decimal places) and one decimal point.
Date	A combination of digits and separators delimited by curly braces {}. The format is determined by the SET DATE and SET CENTURY commands.
Logical	A choice of two values, .T. for true, .F. for false

The TYPE() function can be used to determine the data type of any expression.

uMemvar	TYPE("uMemvar")	Type
[hello]	'C'	Character
'goodbye'	'C'	Character
"RECITAL"	'C'	Character
1234	'N'	Numeric
1234.56	'N'	Numeric
"1234.56"	'C'	Character
{01/01/96}	'D'	Date
{01/01/1996}	'D'	Date
"01/01/96"	'C'	Character
.T.	'L'	Logical
.F.	'L'	Logical
opentags	'P'	Procedure
OMyObj	'O'	Object
Undetermined Type	'U'	Undefined

Memory Variables

Naming

Memory variable names must begin with a letter (A-Z, a-z) or an underscore (-), followed by any combination of letters, digits or underscores. The variable name can be of any length, but only the first ten characters are significant, so these must be unique. The Recital/4GL ignores the case of letters, so `m_var`, `M_VAR`, and `m_VaR` would all be treated as the same memory variable name. The name given to a memory variable has no bearing on the type of data that is, or can be, stored in it. In fact, the type of data stored in a particular variable can be changed at any time, although this needs to be carefully controlled so that inappropriate operations are not attempted. e.g.

```
m_var = 1234
m_var = 'a character value'
? m_var + 100
```

Assignment

Values are assigned to memory variables using the `STORE` command or the equals `=` operator.

```
store 'new value' to cVAR1
cVAR1 = 'new value'
```

Note that the `STORE` command can assign a value to more than one memory variable in a single command.

```
store 'new value' to cVAR1, cVAR2
```

Declaration and Visibility

Values can be assigned to memory variables without those memory variables having been pre-declared, but it is generally accepted that the pre-declaration of memory variables is good programming practice.

Variables can be declared as `PUBLIC`, `PRIVATE` or `LOCAL` and will be initialized as a logical false (.F.).

```
public cVar1
private cVar2
local cVar3
```

`PUBLIC` variables are globally visible: they are accessible and can be changed from any part of an application. If the application is run from the Interactive Prompt, then any public variables can still be accessed even after the application ends. Any variables created at the Interactive Prompt are automatically created as public variables.

`PRIVATE` variables are only visible within the declaring module (program, procedure, User Defined Function) and any modules called by that declaring module. Any variables accessed within a module that are not pre-declared are automatically created as private variables. When the module returns, then all of the memory variables and arrays that were declared by the `PRIVATE` command are released.

`LOCAL` variables are only visible within the declaring module and are released when the module returns. `LOCAL` variables differ from `PRIVATE` variables in that a `LOCAL` variable is not visible to lower level procedures or functions.

Arrays

The Recital/4GL supports the use of both one-dimensional and two-dimensional static arrays. In almost all cases, arrays must be pre-declared before reference can be made to them.

Declaration can be made using the `DECLARE / DIMENSION` commands or the `PUBLIC / PRIVATE / LOCAL` commands. The size of the array can be specified in parentheses or square brackets.

```
private aArr1[10]
public aArr2(10)
declare aArr3[10,9]
```

Values can be assigned to a single element by specifying its element number, or to all the elements in an array.

```
// Assign value to all elements
aArr1 = [hello]
```

```
// Assign value to one element
aArr1[1] = [hello]
aArr3[2,4] = [hello]
```

Arrays are always passed to procedures or functions by REFERENCE i.e. any changes made in the called program will be reflected in the actual array.

Dynamic Arrays

The Recital/4GL also supports the definition of dynamic arrays - arrays to which additional elements can be added. To declare a dynamic array, simply omit the number of elements.

```
private aDynarray[]
```

Elements are added using arrayname.element syntax.

```
aDynarray.name = [Recital Corporation]
aDynarray.email = [info@recital.com]
```

They can then be referenced by element number or by element name.

```
? aDynarray.name
Recital Corporation
? aDynarray[2]
info@recital.com
```

Mathematical Operators

The Recital/4GL supports the use of the following Mathematical Operators:

Operator	Operation	Precedence	Data Types
()	Parentheses	1	N,C,D
**	Exponentiation	2	N
*	Multiplication	3	N
/	Division	3	N
%	Modulus/Remainder	3	N
+	Addition	4	N,C,D
-	Subtraction	4	N,C,D

When dealing with Character data types, the operators have the following definitions:

Operator	Operation
+	Concatenate the right hand string to the end of the left hand string
-	Concatenate the right hand string to the end of the left hand string after trimming the left hand string of trailing spaces

Example

? 2*3^2

18

? 2*25%7

1.00

? [Hello] - [World]

Hello World

Logical Operators

The Recital/4GL supports the following Logical Operators:

Operator	Operation
.AND.	Logical AND
AND	Logical AND
.OR.	Logical OR
OR	Logical OR
.NOT.	Logical NOT
!	Logical NOT
.XOR.	Logical Exclusive OR
XOR	Logical Exclusive OR

Statements containing Logical Operators are automatically optimized in the following way:

If the Left Hand Side (LHS) of an AND statement is false, then the Right Hand Side (RHS) is parsed, but not evaluated, unless the statements are enclosed in parentheses.

If the LHS of an OR statement is TRUE, then the RHS is parsed, but not evaluated, unless the statements are enclosed in parentheses.

The above optimizations can be disabled, causing all entries with the statement to be evaluated, if the optional Environment Symbol DB_OPTLOG is defined as either "OFF" or "NO", prior to starting the Recital product.

The Logical Operators are evaluated from left to right in the following order:

1. Statements enclosed in parentheses
2. .NOT./!
3. .AND.
4. .OR., .XOR.

Example

```
? .T. .and. .F. .or. .T.
.T.
// Default setting of DB_OPTLOG
? .F. .and. .T. .or. .T.
.F.
? (.F. .and. .T.) .or. .T.
.T.
```


Relational Operators

The following Relational Operators are supported in the Recital/4GL:

Operator	Operation
=	Equal To
==	Exactly Equal To / Matches Pattern
<>	Not Equal To
!=	Not Equal To
#	Not Equal To
>	Greater Than
>=	Greater Than or Equal To
<	Less Than
<=	Less Than or Equal To
\$	Substring is Contained In
	Contains Substring

The Relational Operators are always evaluated from left to right.

The following 'wildcard' characters can be used for == pattern matching:

Character	Action
?	Matches any one character
%	Matches any one character
*	Matches zero or more characters

In SQL mode (SET SQL ON or embedded EXEC SQL statements), the following wildcard characters are available:

Characters	Description
_	Matches any one character
%	Matches zero or more characters

Note: For FoxPro compatibility reasons, wildcard pattern matching is not available when SET COMPATIBLE is set to FOXPRO/FOXBASE/FOXPLUS/VFP.

Example

```
cSTR1 = [Welcome to the Recital/4GL]
? "Recital" $ cSTR1
.T.
```

```
cSTR2 = [Welcome]
// Compares to the end of cSTR2
? cSTR1 = cSTR2
.T.
```

```
// Compare contents & size
? cSTR1 == cSTR2
.F.
```

Special Purpose Operators

Three special operators exist within the Recital/4GL. These are:

The Macro Operator (&)

When an ‘&’ ampersand character precedes a variable or an expression contained within parentheses, the result of the expression is substituted into the command. Nested macros are not supported.

The Alias Operator (->)

An open table can be referred to by its alias name and its fields can be accessed using the alias operator. The alias name is either a name you have specified in the USE <table> ALIAS <alias name> command, or, by default, the first ten characters of the table basename. The letters a-z (excluding m) can also be used as an alias to the work areas 1-26 (excluding 13). M is used to reference memory variables, so is not available as a table alias

The Dot Operator (.)

The dot operator ‘.’ is used to reference either the properties of objects, or it can be used interchangeably with the alias operator.

Example

```
// Macro Operator Example
cTABLE = [employees]
use &cTABLE
```

```
// Alias Operator Example
cEMPNAME = &(cTABLE + "->NAME")
```

```
// Dot Operator Example
use employees
```

```
class EmpRecord
public:
    property NAME
    property SALARY
public:
    method IDENT
    return "Employee Record"
endclass
```

```
oMYREC = new EmpRecord()
cTYPE = oMYREC.IDENT()
? cTYPE
```

Recital/4GL Basics

Recital/4GL statements can be entered in a free-form fashion. Spaces and indents can be added where required to make the code more readable. Commands can be entered in upper, lower or mixed case and in almost all instances, only the first four letters of a command need be entered, e.g.

```
MODIFY STRUCTURE
MODI STRU
SET EXCLUSIVE ON
SET EXCL ON
```

Comments

Comments are ignored by the compiler but are, of course, very useful to the developer and even more so to anyone who has to maintain or alter the code later in its lifetime.

Symbol	Position	Effect
*	Start of line	Line is ignored
//	Anywhere in line	All text that follows is ignored
&&	Anywhere in line	All text that follows is ignored

Line Continuation

The semi-colon “;” placed at the end of a line can be used to continue a command onto the next line. Commands can be up to 8192 characters long.

```
use customer;
    in 1;
    order customer
```

Is the equivalent of:

```
use customer in 1 order customer
```

Multiple Commands

By contrast, a semi-colon within a line signifies the end of a command and that the text that follows is a new command.

```
select 1; use customer; set order tag customer
```

Is the equivalent of:

```
select 1
use customer
set order tag customer
```

Parameter Passing

By VALUE / By REFERENCE

Parameters can be passed to procedures, programs and UDFs (User Defined Functions) by VALUE or by REFERENCE. The called module must have a PARAMETERS statement at the top of the code.

e.g.

```
PROCEDURE MyProc
PARAMETERS cPARA1
//...
//...
RETURN
```

When a parameter is passed by VALUE, a copy of the memory variable is passed to the module and the original memory variable is not accessible within the called module.

When a parameter is passed by REFERENCE, the called module is given the address of the memory variable so the memory variable itself can be altered.

Function calling syntax passes parameters by VALUE by default.

MyUdf(cVAL)

To pass by REFERENCE, the parameter should be preceded by an @ sign.

MyUdf(@cREF)

Procedure calling syntax passes parameters by REFERENCE by default.

do MyProc with cREF

To pass by VALUE, the parameter must be enclosed by parentheses.

do MyProc with (cVAL)

The PARAMETERS() function and the PCOUNT() function return the number of parameters passed to a module.

Parameter Passing from the Operating System

When Recital Terminal Developer programs are run from the Operating System, up to nine parameters may be passed to the program (compiled .dbo or non-compiled .prg). No PARAMETERS statement is required within the program, since the arguments will be stored in automatically declared public memory variables named _para1 to _para9.

\$ dbrt myapp "one" "two" "three" "four" "five" "six" "seven" "eight" "nine"

Default File Extensions

Extension	File Type
.brg	BRIDGE
.cat	CATALOG
.clo	COMPILED CLASS LIBRARY
.cls	CLASS LIBRARY
.col	COLOR SCHEME DEFINITION
.dbd	DICTIONARY
.dbf	TABLE
.dbj	AFTER IMAGE JOURNAL TABLE
.dbj	AFTER IMAGE JOURNAL MEMO
.dbo	COMPILED SOURCE CODE
.dbt	MEMO
.dbx	MULTIPLE INDEX
.def	TERMINAL DEFINITION
.dkf	DES3 ENCRYPTION KEY
.dtd	DOCUMENT TYPE DEFINITION
.fmo	COMPILED FORMAT
.fmt	ASCII CUSTOM FORMAT
.frm	REPORT WRITER REPORT
.gtw	GATEWAY
.his	HISTORY TRACE
.hlm	HELP MENU
.hlt	HELP TEXT
.img	SCREEN IMGE
.kgw	GATEWAY DEFINITION
.kqy	SQL QUERY
.lbl	LABEL FORMAT
.log	BEFORE IMAGE JOURNAL
.mdf	MENU DEFINITION
.mem	MEMORY VARIABLES / ERROR FILE
.ncs	NATIONAL CHARACTER SET
.ndx	SINGLE INDEX
.qry	QUERY
.prg	SOURCE CODE
.scr	FORMS DESIGNER BINARY
.sdb	SCHEDULE TABLE
.sdt	SCHEDULE MEMO
.sdx	SCHEDULE INDEX
.src	LINKED SOURCE
.str	STRUCTURE TABLE
.trf	TREPORT REPORT
.txt	ASCII TEXT
.vue	VIEW
.win	WINDOW DEFINITION
.xml	XML

Commands invoked from the Operating System

The following commands for Recital Terminal Developer are invoked from the Operating System.

db [-p <license text file>] | [-r] | [-w] [-q] [-f] [-x <compiled program>]

Starts Recital Terminal Developer in development mode. In development mode, the command window or prompt and the development tools are accessible. Development mode access requires a valid development license.

-p	Loads the information from a license text file into the Product Registration Screen, and then allows the license to be saved before starting.
-r	Displays the Product Registration Screen, allowing license details to be added or amended before starting.
-w	Start without the default windows active. The default windows are defined in the main config.db.
-q	Start without the default windows active and without showing the license information screen.
-f	Start in Fox compatibility mode. The following are active: SET COMPATIBLE TO FOXPRO and SET FILETYPE TO FOXPRO. This can also be achieved using the <i>fox</i> script instead of the <i>db</i> script.
-x	Execute the specified program.

dbdemo

Starts Recital Terminal Developer in Assistant mode in the demo directory.

dbl -i <input file> -o <output file> -x <first procedure> -m <message file>

Calls the Recital/Linker to link program source. Please see the Recital/Linker documentation for full details.

dbrt <compiled program>

Runs the specified <compiled program> in runtime mode. In runtime mode the development tools are not accessible and all source files must be compiled. For more information on compiling, please see the COMPILE command. Runtime mode access requires a valid runtime license. The following alternative syntax can also be used:

db [-q] -x <compiled program>

dbstop <pid>

Cleanly stops the Recital process identified by the process identification number in <pid>.

dbusers -c | -l | -r

Creates, resets or lists the users file or files. Please see the Users Files documentation for full details.

dbvideo

Starts the Recital interactive feature demonstration.

The Recital/Linker

Recital Terminal Developer includes the Recital/Linker utility, which can be used to link individual program files into a single file that can then be compiled. This limits the number of files that need to be open at one time, thus using less OS file handles and making more efficient use of shared memory when running applications with SET PSHARE ON.

With SET PSHARE ON, compiled programs are loaded into shared memory when called. All users accessing a particular program can access the same area of shared memory rather than loading the program into private memory. The program is removed from shared memory when it no longer has any attached users. A single compiled program therefore, need only be loaded once and accessed by all users. Multiple smaller compiled programs cause an increased amount of loading and unloading activity in shared memory.

The linker is run from the Operating System. You must specify the name of an input file and an output file. You can optionally specify the first procedure to be run and the name of a file to which all linker messages will be sent.

```
$ dbl -i <input file> -o <output file> -x <first procedure> -m <message file>
```

Input File

The input file should contain the name of each procedure to be linked. These should be listed one to a line and must be unique e.g.

```
prog1.prg
prog2.prg
prog3.prg
```

Output File

The output file is the file that will be compiled and run. To distinguish this from the .prg files from which it is comprised, the convention is to give this file a .src extension.

First Procedure

If the first procedure to be run is specified, a 'do <first procedure>' line will be included in the output file. The file can be run as a self-contained module rather than being used as a procedure library.

Message File

If no message file is specified, the messages will be displayed on the screen (standard output) while the linker is running. The name of each file is listed as it is linked and a warning message is given if any duplicate names are found.

Compiling large files

When you come to compile your .src file, the linker output file, you may find the following error message is generated part way through the compilation:

```
Maximum compile file size (MAXDBO) of <value> exceeded with file <.src file>.
```

If this occurs, you need to increase the SET MAXDBO TO <expN> value. This setting controls the size of compiled file that can be created. The <expN> needs to be 4x the size of the compiled file in kilobytes. The default is 256.

SET MAXDBO TO <expN> is set in the main config.db file, but can also be issued at the interactive prompt.

Invoking the Recital/Linker from the Recital/4GL

The Recital/Linker can also be invoked through the Recital/4GL using the LINK command.

```
LINK FROM <.dbl file> | <skeleton> TO <.src file>  
[MESSAGE <.map file>]  
[COMMAND <expC>]  
[COMMENTS]
```

Please see the LINK command for full details.

DBUSERS

Class

Utilities

Purpose

Recital Terminal Developer utility to reset or create the users files or to display the currently active users

Syntax

`dbusers -c | -l | -r`

See Also

DISPLAY USERS, LIST USERS

Description

The DBUSERS utility is used to reset or create the users files or to display the currently active users. When Development users log in, their details are logged in the users.db file. When Runtime users log in, their details are logged in the rtusers.db file. When a user logs out, their details are removed from the appropriate users file. The users files are created when the license is installed using 'db-r' or 'db -p'. If the license file is created manually, the users files are not created automatically. To manually create the users files, the following command should be issued at the Operating System prompt:

```
dbusers -c
```

Please note, that the users files are created based on the license file – users.db is only created if the system is licensed for Development users, rtusers.db if the system is licensed for Runtime users.

If the users files become out of sync with the actual users logged in, for example if a connection is broken and the session is not closed down cleanly, the users files can be reset. All users must exit the Recital product. To reset the users files, the following command should be issued at the Operating System prompt:

```
dbusers -r
```

From within the Recital environment, the DISPLAY USERS or LIST USERS commands can be used to view the users logged in to the current users file (rtusers.db if the command is run in the Runtime environment, users.db if in the Development environment). To view current Development and Runtime users from the Operating System, the following command should be issued at the Operating System prompt:

```
dbusers -l
```

Example

```
# dbusers -r
```

Recital: Users file reset.

Recital: Runtime users file reset.

Products

Recital Terminal Developer

DBEXEC

Class

Utilities

Purpose

Used to run program files in the background via the Recital Database and Mirage Servers

Syntax

```
dbexec <program-file> [<_para1> <_para2> ... <_para9>]
```

Description

The DBEXEC utility is used to run program files in the background via the Recital Database Server or Recital Mirage Server. It gives the ability to run source code directly on the server without any client connection in place. The DBEXEC command should be issued from the Operating System prompt and followed by the name of the program file to be run. Parameters can be passed to the program by specifying them after the program name. When specifying more than one parameter, white space should separate each parameter. A maximum of nine parameters can be passed. The parameters are stored in predefined variables _para1 to _para9 and no parameters statement is required in the program itself.

Example

```
# dbexec maint_prog "customers" "contacts"
```

Products

Recital Database Server, Recital Mirage Server

Printing

The printing commands within the Recital/4GL allow you to send output to a system printer, to a file or to a local printer attached to the printer port of your terminal.

Printing to a System Printer

When you have a default printer destination set on your system, the following command will send the specified file to that printer:

```
print <filename>
```

To redirect output to a system printer, the command SET PRINTER TO \\SPOOLER is used. This picks up the environment variable DB_PRINT (set in profile.db). The value of DB_PRINT must be either a valid Operating System print command, e.g. on UNIX:

```
DB_PRINT="lp -dmyprinter -onobanner"
```

or the name of a file which contains such a command (and any other processing required.) By default, DB_PRINT is set to the file 'print.<os>' in the product home directory, e.g. "print.unix" in the UD directory. This can be modified if required, to suit your particular printing environment. The 'print.<os>' file contains a further environment variable, called DB_PRINTOPT, which can be set to the options that you require on the OS print command. By default, the value of DB_PRINTOPT is set in profile.db, but the PUTENV() function can be used to change this value, and the GETENV() function to query it. The following commands will cause an environment status listing to be printed according to the command(s) specified by DB_PRINT:

```
set printer to \\spooler
list status to print
set printer to
```

The following will send output to the system printer (and the screen):

```
set printer to \\spooler
set print on
&& ...output
set print off
set printer to
```

Stopping Output to the Screen

To send output solely to the specified printer, and not to the screen, the following commands are required:

```
set screenmap off
set console off
set device to print
```

And to return control to the screen:

```
set device to screen
set console on
set screenmap on
```

Printing to a File

For printing to a file, the same commands apply, but you should specify the filename in the SET PRINTER TO command:

```
set printer to myfile.txt
list memory to print
set printer to
```

Printing to a Slave Printer

If you have a printer attached to the printer port of your terminal or PC, the SET PRINTER TO command is not required. Use the SET PRINT ON / OFF command to redirect output to the printer. To prevent output also being sent to the screen, use the commands as listed above. To switch your terminal output to screen / printer or printer only mode, the control sequences specified in entries 4, 5 and 6 of the Terminal Definition File are used.

```
// Program to send display to local printer
```

```
set screenmap off
set console off
set device to print
set print on
@1,1 say [hello world!]
set print off
set device to screen
set console on
set screenmap on
```

Sending Control Sequences

Control sequences can be sent to your printer using the ? or ?? commands. The command SET SCREENMAP OFF must be issued before sending the sequences to prevent them being re-mapped, e.g.

```
// Program sending control sequences and text to a file
```

```
set printer to myfile.txt
set screenmap off
set console off
set print on

// Switch into 'emphasis mode'
? CHR(27) + "E"
?? [HELLO]
// Switch out of 'emphasis mode'
? CHR(27) + "F"
?
? [HELLO AGAIN]
set printer to
set print off
set console on
set screenmap on
```

Control sequences sent must be appropriate to your printer.

PRINTSCREEN()

The PRINTSCREEN() function can be used to dump a copy of the current screen to a file or to the printer. The function is most often called from within a hot key procedure to enable screen dumps of full screen forms, e.g.

```
// Hot key procedure
procedure pscreen
set printer to \\spooler
printscreen()
set printer to
return
```

```
// Current screen is sent to system printer when [TAB] key is pressed
on key label tab do pscreen
```

Printing on Windows

Since Recital Database Server processes are not GUI based, print jobs on Windows systems can only be sent to printers configured for DOS access. The required DOS print command should be specified using the command SET PRINTER TO <OS command>.

For Recital Mirage applications, please see the PRINTFILE() function.

Database Table Encryption

From Recital 8.5 onwards, Recital installations that have the additional DES3 license option have the ability to encrypt the data held in Recital database tables. Once a database table has been encrypted, the data cannot be accessed unless the correct three-part encryption key is specified, providing additional security for sensitive data.

For more information on using DES3 Encryption, please see the following commands:

- DECRYPT
- ENCRYPT
- SET ENCRYPTION

Encryption also affects the following commands:

- APPEND FROM
- COPY FILE
- COPY STRUCTURE
- COPY TO
- DIR
- USE
- SQL INSERT
- SQL SELECT
- SQL UPDATE

Form & Worksurface Function Keys

The Function Keys within the Recital user interface have specific actions in the screen forms and in the Recital Terminal Developer Development Tools. Each Function Key also has a matching 'alternate key', which is a combination of the Control key and a letter.

In the Recital Terminal Developer Development Tools, a Key Help Panel appears at the top of the screen, listing the key to use for functions within that Work Surface. Press the [TAB] key to display the Key Help Panel if it is not displayed. The keys and their labels are taken from the Terminal Definition file; section 5 entries 114 to 139.

The caret ^ in the alternate key sequence indicates that the [Ctrl] control key and the key that follows should be pressed simultaneously.

Name	VT Keyboard	PC Keyboard	Alternate
HELP	Help	F1	^C
ABANDON	F11	F11 / Esc	^G
EXIT/SAVE	F20	End	^W
MENUBAR	Do	Home	/
NEXT RECORD/PAGE DOWN	Next	PgDn	^N
PREVIOUS RECORD/PAGE UP	Prev	PgUp	^R
FIND	Find	F9	^K
FIND NEXT	Select	F10	^L
DELETE RECORD	F19	F5	^U
DELETE CHAR	F12	Del	^H
DELETE FIELD	Remove	F8	^Y
TAB	Tab	Tab	^I
PAN RIGHT	F9	F7	^B
PAN LEFT	F10	F6	^Z
UPDATE MODE	F17	F4	^T
REFRESH	F18	F12	^D
CARRY MODE (APPEND)	F17	F4	^T
INSERT	Insert	Ins	^V
WORD LEFT	F13	F2	^A
WORD RIGHT	F14	F3	^F

National Character Sets

To enable the use of National Character Sets, the Recital 4GL supports the following functionality:

- Alternative collating sequences for indexes and sorts
- The input of 8 bit characters.

These are achieved using the SET LANGUAGE TO <language-keyword> command and in the case of alternative collating sequences, the COLLATE() function.

On execution of the SET LANGUAGE TO command, the appropriate National Character Set file is loaded from the 'terminals' directory. The following <languages-keyword> values are supported:

- AMERICAN
- BELGIAN
- BRITISH
- CHINESE
- DANISH
- DUTCH
- FINNISH
- FLEMISH
- FRENCH
- GERMAN
- ITALIAN
- JAPANESE
- NORWEGIAN
- RUSSIAN
- SPANISH
- SWEDISH

Please note that the <language-keyword> specified is used to reference the appropriate National Character Set file and not to 'switch' to a particular language. If the <language-keyword> is any of the above, apart from AMERICAN or BRITISH, then 8 bit characters supported by the current terminal will be enabled. For example, SET LANGUAGE TO FRENCH will still allow Russian or German specific characters supported by the current terminal to be entered.

The National Character Set files reside in the 'terminals' directory and have filenames equivalent to:

<terminal-type>-<first-3-characters-of-language>.ncs

For example, the GERMAN National Character Set file for the VT200 terminal is vt200-ger.ncs.

If the National Character Set file is being used solely to enable 8 bit characters, its contents are disregarded and it can in fact be empty.

To enable alternative collating sequences for indexes and for the SORT command, the National Character Set file must contain the appropriate collating sequence. This consists of ASCII value pairs, the first being the ASCII value of the character, the second its position in the collating sequence, e.g.

97=65
65=66
224=67
192=68
225=69
193=70
226=71

Indexes and sorts must use the COLLATE() function to make use of this alternative collating sequence, and the SET LANGUAGE TO <language-keyword> must be in effect before the index or sort operation is carried out and at all times when any such indexes are being used. To give system wide National Character Set support, the SET LANGUAGE TO <language-keyword> should be issued in the config.db file.

Extensible Markup Language (XML)

Extensible Markup Language allows data and data structures to be defined in an agreed format so that they can be shared. The Recital 4GL and Recital SQL provide commands to export Recital data in XML format and import XML data into Recital format. The Recital 4GL also includes a series of functions for accessing XML files and their associated Document Type Definition (DTD) files.

Format of XML files

The format for XML files can be either RECITAL or ADO (Microsoft® ActiveX® Data Objects). Any XML files created in the ADO format can be loaded with the Open method of an ADO Recordset object. The default XMLFORMAT setting is ADO. The default XMLFORMAT setting can be changed using the SET XMLFORMAT command or overridden using the FORMAT clause on the SQL SELECT statement.

Note: The XMLFORMAT setting determines whether Recital creates an accompanying DTD file when creating XML files. A DTD file is only created when XMLFORMAT is set to Recital.

- SET XMLFORMAT - Toggle XML file format (RECITAL/ADO).

Exporting Recital Data in XML Format

- COPY TO ... TYPE XML - The COPY TO command allows records from the active table to be copied out in XML format (RECITAL/ADO).
- SELECT ... SAVE AS XML - The Recital/SQL SELECT command allows data from one or more tables or views to be saved in XML format (RECITAL/ADO).
- FETCH ... INTO XML - The Recital/SQL FETCH command allows individual rows from a SELECT statement to be saved into a RECITAL XML format file.

Importing XML Data into Recital Format

- CREATE TABLE...FROM XML - The Recital/SQL UPDATE command allows a Recital table to be created from a structure or structure and data from an ADO XML format file.
- INSERT...FROM XML - The Recital/SQL UPDATE command allows a Recital table to be updated with data from an ADO XML format file.
- UPDATE ... FROM XML - The Recital/SQL UPDATE command allows a Recital table to be updated with data from a RECITAL XML format file.

Functions for Accessing XML and DTD files

- XMLCOUNT() - Function to return the number of records from an XML file.
- XMLCREATEDTD() - Function to create a Document Type definition file for a particular table.
- XMLFIRST() - Function to read the first record contained in the specified XML file.
- XMLNEXT() - Function to read the next record contained in the XML file specified with the XMLFIRST() function.
- XMLVALIDATE() - Function to return the number of records from an XML file.

Error Handling & Debugging

Error Handling

At runtime, the Recital/4GL will stop execution display an error message and create an error.mem file whenever an error occurs in the application, unless an alternative error handling procedure has been specified.

The error.mem file includes the following information:

- The Recital Product version, release, platform and compile date information
- System information, such as maximum file size and maximum process stack
- Recital License information
- User and Node names and process ID
- The error number
- The error message
- The program line
- The program line number
- A stack trace of the programs and procedures called
- A list of all active public and private memory variables and classes
- Status of open tables, indexes and current records.
- Settings listing as per LIST STATUS

The error.mem file will be named *error.mem* and will be overwritten by subsequent errors unless SET ERRORVERSION is ON. If SET ERRORVERSION is ON, multiple numbered error.mem files are created with the following naming format:

```
error0001.mem
error0002.mem
error0003.mem
```

The error.mem file(s) will be created in the directory that is current when the error occurs unless the DB_ERRORDIR environment variable/symbol is set. If set, DB_ERRORDIR points to a directory in which the error.mem file(s) will be created.

Alternative error handling procedures can be specified using the ON ERROR command. When there is an active ON ERROR setting, no error.mem file will be created and program execution will not be halted. It is now up to ON ERROR error handler to trap the information required in order to trace the error and to take appropriate action based on the error that has occurred.

Note: The SAVE ERROR command allows for the creation of an equivalent of the error.mem file. Several functions are available to give information about errors:

FUNCTION	RETURNS
ERRNO()	Operating System Error Number
ERROR()	Recital 4GL Error Number
MESSAGE()	Error Message
MESSAGE(1)	Line which caused the error
PROCLINE()	Currently executing procedure line number
PROCNAME()	Currently executing procedure name

It is important to note that the PROCLINE() and PROCNAME() functions return information about the CURRENTLY executing procedure. If these functions are called from within the error handling procedure, they will give information based on the error handling procedure itself and not the procedure in which the error occurred. They should, therefore, be specified as parameters to the error handler, e.g.

```
on error do MyErrProc with procname(), procline()
```

Debugging

In Recital Terminal Developer, the DEBUG command displays a pop-up debugger allowing the specified program to be monitored during execution. The pop-up debugger can also be activated by using the SET STEP ON command. To debug an entire program, issue SET STEP ON before starting. To debug just a particular section of the program, insert the command into the code. The Debugger can be toggled on and off while running your program by means of a hot key procedure, e.g.:

```
procedure SetStep
// set step to opposite of current value
set step (!(set([step])))
return

procedure DefineKey
// set [TAB] as the hot key to call SetStep
on key label tab do SetStep
return
```

During the debugging process, the program must be running in interpreted mode, not in compiled mode. The DEBUG command will do this automatically. To ensure that the program is running in interpreted mode when using SET STEP ON/OFF, issue the following two commands before starting program execution:

```
SET COMPILE OFF
SET DEVELOPMENT OFF
```

When running in interpreted mode, a complete history trace of all commands executed can be generated. This requires the use of the SET HISTORY and SET DOHISTORY commands.

Example

```
set compile off
set development off
set history on
set dohistory on
set history to file myhis
use demo
edit
set history to
set dohistory off
vi myhis.his
```

Recital/4GL Error Messages by Number

ERROR()	Error Message
1	Line exceeds maximum width of <expN> characters
2	Syntax Error in command
3	Illegal character in command line
4	Unterminated string
5	String is too long
6	Name is too long
7	Insufficient memory
8	Boolean operator was expected
9	DOs nested too deep
10	Unrecognized phrase/keyword
11	Filename was expected
12	File cannot be found or is not executable
13	This command/feature is not available in RECITAL
14	File already exists
15	Too many files are open [errno() is 24]
15	File does not exist [errno() is 2]
15	File access permission denied
15	File is in use by another user [errno() is 11]
15	Cannot open <filename> - error <error #>
16	No database is in USE
17	Invalid filename
18	Number was expected
19	Maximum files allowed open is <maximum number of workareas>
20	Value is out of range
21	Record is out of range
22	Syntax error in expression
23	Expression is too complicated
24	Fatal system error
25	Keyword TO was expected
26	Variable name was expected
27	Data type mismatch
28	Keyword RETURN or end of file was expected
29	Bad statement nesting or command used out of context
30	Logical expression was expected
31	Cannot erase <filename>
32	Skeleton name was expected
33	Unexpected end of file encountered
34	Memory variable was expected
35	Memory variable does not exist
36	Syntax error
37	PROCEDURE name was expected
38	Too many @...GETs defined
39	Invalid screen co-ordinates
40	Invalid picture
41	String was expected
42	Variable/field <variable/field name> not found
43	Field variable was expected
44	Database <database name> is already open
45	Keyword WITH was expected

46	Expression was expected
47	Keyword FROM was expected
48	Keywords ON or OFF were expected
49	Field variable is not in currently selected database
50	Keyword FOR or WHILE were expected
51	Comma was expected
52	Record number is out of range
53	Help text for GET is too long
54	RANGE can only be checked on dates and numerics
55	Invalid LOOKUP field
56	Keyword FILE was expected
57	Keyword BY was expected
59	No such window
60	File is not of recognizable format
61	Invalid key
62	Database is not indexed
63	Invalid parameter
64	File <filename> already exists
65	Keyword ON was expected
66	Keyword VIA was expected
67	Database is not indexed
68	Keyword INTO was expected
69	ALIAS name was expected
70	Illegal data format in FROM file
71	Too many SUM expressions, maximum is <maximum # of sum expressions>
72	'(' was expected
73	')' was expected
74	Data type mismatch
75	Value should not have any decimal places
76	FOR was expected
77	Invalid WITH file
78	Invalid join field
79	Too many fields in field list
80	Keyword FIELD was expected
81	Database is already open
82	No such index
83	Database is not shareable
84	Field definitions in view file too long
85	Keyword REPLACE was expected
86	Maximum database records exceeded
87	Invalid date
88	Syntax error in terminal definition file
89	LOCK failed
90	UNLOCK failed
91	Incorrect use of reserved word
92	PROCEDURE name already in use
93	Relation expression too long to save
94	Internal Error - View Workarea Number
95	Filter expression too long to save
96	Cannot run <filename> - errno <error #>
97	Unrecognized command verb
98	Too many strings - increase STRINGBUF
99	INDEX key is too long, maximum length is <expN>

100	Maximum accumulators exceeded
101	Syntax error in column[<column #>] / subtotal[<subtotal #>] expression
102	Command not available with RMS indexed sequential files
103	Required facility is not available with this license
104	ALIAS name already in use
105	Lock failed - database not available
106	No assistance available
107	Index file does not match database
108	Error text for GET is too long
109	IN was expected
110	Cannot replace as key already exists
111	Exclusive use of database is required
112	Unrecognized function name
113	Database <database name> locked by another user
114	Mismatched IF.ELSE.ENDIF
115	Mismatched DO WHILE and ENDDO
116	Cannot lock database - exceeded enqueue quota (ENQLM)
116	Cannot lock database - errno <error #>
117	Cannot lock index - exceeded enqueue quota (ENQLM)
117	Cannot lock index - errno <error #>
118	Cannot lock record - exceeded enqueue quota (ENQLM)
118	Cannot lock record - errno <error #>
119	Lock table is full
120	Record not in index
121	Too many SORT fields, maximum is <maximum sort fields>
122	Cannot rename <filename> - errno <error #>
123	Illegal use of SUSPEND
124	<Reserved>
125	Cannot update record - errno <error #>
126	Cannot delete record - errno <error #>
127	Cannot append record - errno <error #>
128	Too many relationships - maximum is <maximum workareas - 1>
129	Related database is not indexed
130	Cyclic relation
131	Database access permission is Read Only
132	Key field does not exist in both databases
133	You cannot update a database from itself
134	Too many lines in footer block
135	Too many lines in header block
136	FORMAT file is active
137	Fatal I/O error reading index file - exceeded quota
137	Fatal I/O error reading index file - errno <error #>
138	Fatal I/O error writing to index file - exceeded quota
138	Fatal I/O error writing to index file - errno <error #>
139	Fatal I/O error reading from database file - exceeded quota
139	Fatal I/O error reading record <rec #> from database file - errno <error #>
140	Fatal I/O error writing to database file - exceeded quota
140	Fatal I/O error writing record <rec #> to database file - errno <error #>
141	Variable name already in use as a PROCEDURE
142	SUBSTR() start point out of range
143	Variable/field <variable/field name> already exists
144	Unrecognized bridge type
145	Cannot execute as replay mode is already active

146	Cannot execute as capture mode is already active
147	Cannot unlock database - exceeded enqueue quota (ENQLM)
147	Cannot unlock database - errno <error #>
148	Not suspended
149	Valid only in programs
150	Syntax error in browse character expression
151	Memo (.DBT) file cannot be opened
152	Record is in use by another user
153	Dictionary (.DBD) file cannot be opened
154	Dictionary (.DBD) file cannot be created
155	TO clause must be specified
156	ON clause must be specified
157	ARRAY reference is out of bounds
158	ARRAY not found
159	']' was expected
160	ARRAY name was expected
161	'[' was expected
162	Illegal use of ARRAY
163	Internal Error - <VMS Operating System Error Message>
164	Internal error - <error #>
165	Validation on field <field name> failed
166	Syntax error in field <field name> expression
167	Calculated field <field name> cannot be updated
168	Incompatible data type for field <field name> default
169	<Reserved>
170	Syntax error in field <field name> default expression
171	Incompatible data type for field <field name> default
172	Incompatible data type in calculation expression for field <field name>
173	Syntax error in calculation expression for field <field name>
174	Position is off the screen
175	Keyword was expected
176	Bad keyword
177	FUNCTION name already in use
178	FUNCTION name was expected
179	Function <function name> did not return a value
180	Illegal use of MEMO field
181	Not a RECITAL database
182	Record is out of range
183	Internal locking error in 'lock_current_record()'
184	Too many screens saved. Maximum is <maximum saved screens>
185	Screen memory variable was expected
186	Invalid workarea specified
187	Command not available with this type of bridge
188	This key cannot be redefined
189	Nested transactions are not supported
190	Cannot open transaction log file <log file>
191	Command illegal when transaction in progress
192	Transaction log file I/O error
193	Internal transaction log error
194	TRIM functions cannot be used in index expressions
195	Database structure and RMS record length do not match
196	Too many @...MENU's defined
197	Procedure/Function <procedure name/function name> not declared

198	Data type mismatch in parameter list
199	SQL functions are only valid in SQL commands
200	Nested SQL functions are not supported
201	Invalid parameters for COUNT()
202	Required facility is not available with this license
203	Keyword AT expected
204	Maximum file size (ulimit) of <max file size> exceeded
205	Shared files are not available with a single user license
206	Array name <array name> conflicts with function name
207	Unrecognized Treport (RDL) directive
208	Language definition file not available for this terminal
209	Conflicting keywords/clauses or dictionary entry for @...GET
210	Specified keyword/clause cannot be used with a MEMO in @...GET
211	SORT key is too long, maximum is <maximum index length>
212	MENU name was expected
213	Duplicate MENU name
214	PAD name was expected
215	MENU not defined
216	Keyword PROMPT expected
217	Duplicate PAD name
218	POPUP name not specified
219	Duplicate POPUP name
220	POPUP not defined
221	BAR position exceeds menu size
222	BAR position already in use
223	BAR definition invalid on this type of POPUP
224	Keyword OF expected
225	PAD not defined
226	Keyword MENU expected
227	Nested CALCULATE functions not supported
228	Too many CALCULATE expressions, maximum is <maximum number of sum exp.>
229	File is already open
230	Invalid function parameter
231	No current record - database is empty
232	No current record database at EOF
233	You do not have authorization to perform this operation
234	You do not have authorization to modify this field
235	Too many PROCEDURE libraries
236	Get memory variable expected
237	Too many paths specified. Max is 10
238	Character expression was expected
239	Number is too large
240	File is not a valid RECITAL report format
241	WINDOW name was expected
242	WINDOW already declared
243	WINDOW not defined
244	Bad color specification
245	Table name expected
246	Table already declared
247	Table not declared
248	Too many parameters. Maximum is 40.
249	No such workarea. Range is 1 to <Max # of workareas>
250	Keyword "MEMO" expected

251	<Remote server error message>
252	Distributed cache failure - exceeded enqueue quota (ENQLM)
253	Attempt to UPDATE an unlocked record
254	Operation not allowed with this type of server
255	Variable has undefined contents
256	A database is already active in this workarea
257	RSI is already active in this workarea
258	Cannot establish gateway to server <servername>
259	Unrecognized type of network protocol
260	Keyword "DATABASE" expected
261	No client/server gateway active
262	Operation not allowed with this type of gateway
263	Menu memory variable was expected
264	Cannot connect to <servername> server on <machine name> host machine
265	Fatal lseek() error on index file - <index filename>
266	MEMO field expected
267	WINDOW is too small to contain a TITLE
268	FORMAT file is not active
269	Invalid SET expression
270	TAG name was expected
271	Multiple index file <index filename> is not open
272	Cannot create TAG <tag name>
273	Index TAG <tag name> already exists
274	Too many indexes, maximum is 20
275	TAG not found
276	Multiple index file not open in current work area
277	Cannot open multiple index file <index filename>
278	No more work areas available
279	This database already has a dictionary
280	Invalid file structure
281	BAR not found
282	Expressions and TO variables do not match
283	Keyword PRODUCTION was expected
284	Mismatched ')'
285	INDEX key expression is too long, maximum length is 511
286	Exceeded file limit while appending record <record #>
287	<MPE TurboImage database specific error message>
288	Database <table name> is opened <file I/O mode> in another workarea
289	Fatal System Error - PACK of RDD table failed
290	Fatal System Error - ZAP of RDD table failed
291	Command/Function is not supported on MSDOS
292	PROPERTY not found
293	Objects and dynamic arrays cannot be redefined
294	Objects cannot be redefined
295	Operation not allowed with objects or dynamic arrays
296	Object not found
297	CLASS name was expected
298	Duplicate CLASS name
299	PROPERTY name was expected
300	CLASS not found
301	METHOD name was expected
302	'::' was expected
303	METHOD not found

304	PROPERTY <property name> already exists
305	METHOD declared EXTERNAL but not defined
306	Attempt to assign incompatible data type
307	Access to LOCAL member of object denied
308	Access to PRIVATE member of object denied
309	Too many CLASS libraries defined. Maximum is 20
310	STATIC variable <variable name> in CLASS <class name> not found
311	Unknown System Function in CLASS definition
312	Illegal System OBJECT operation
313	Unknown value returned from SYSTEM object
314	TAB not declared
315	TAB name was expected
316	Invalid file pointer used
317	No primary key has been specified on server table
318	Cannot lock global memory - errno <error #>
319	Error accessing data table, error number 319. Contact Recital support
330	Invalid VERSION/REVISION id
331	Invalid PRODUCT id
332	Invalid authorization code
333	Unauthorized software license, Invalid RDD type
334	Unauthorized software license, Invalid SERVER type
335	Unauthorized software license, Invalid VAX license
336	Unauthorized software license, Invalid CPU type
337	Number of users has been exceeded maximum of <max # of users>
338	Too many REPOSITORY libraries defined. Maximum is 10
339	Could not install NetObject <object name>
340	Universal Object Exchange not active on <node name>
341	This command/feature is not available with this product
342	Warning level set to fatal, trace with iologging
343	Keyword AND expected
344	Keyword NULL expected
345	Keyword AS expected
346	Trial version maximum record limit of <expN> exceeded
347	Database must be converted for version 8.0.
348	Illegal use of GRAPHIC field
349	Dictionary (.DBD) file is corrupted
350	Directory not found '<path name>'
351	Unable to load mail
352	Unable to send mail message
353	Could not install COM object
354	Could not install JAVA object
355	Could not install CORBA object
356	Object type not supported
357	COM method error
358	COM get property error
359	COM set property error
360	JAVA method error
361	JAVA get property error
362	JAVA set property error
363	DLL error
1000	User defined error message

Recital Database and Mirage Server Error Messages

Number	Error Message
-1	Incorrect version in license
-2	Incorrect product code in license
-3	Invalid authorization code in license
-4	Expired license
-5	The host name/TCP/IP address in the license does not match the system it is running on
-6	A connection request was made from a different system and the current license is not an Enterprise Edition
-7	The license file cannot be found.
-8	The license file cannot be opened.
-9	The license file cannot be opened for reading.
-10	Invalid option string in the license.
-11	Client is not licensed to use the Server in the option string.
-12	The number of CPU's is greater than the license allows.

Recital/SQL Error Messages

ERROR()	Error Message
5000	Unrecognized phrase/keyword near column <expN>
5001	Table name was expected
5002	Column name was expected
5003	Column name already exists
5004	Data type was expected
5005	Unrecognized data type for column <column name>
5006	Invalid specification for column <column name>
5007	Keyword PRECISION was expected
5008	Cannot create table <table name>
5009	Invalid table specification
5010	Cannot open table <table name>
5011	Keyword TABLE expected
5012	Keyword ADD expected
5013	<command expression> - operation aborted
5014	Keyword INTO expected
5015	Keyword VALUES expected
5016	Keyword SQL expected
5017	Keyword FROM expected
5018	'=' expected
5019	Keyword SET expected
5020	Keyword INDEX expected
5021	Index name expected
5022	Keyword ON expected
5023	Keyword BY expected
5024	Keyword UPDATE expected
5025	FROM clause missing
5026	Bad ORDER BY clause
5027	Bad GROUP BY clause
5028	Cursor name was expected
5029	Cursor already declared
5030	Keyword CURSOR was expected
5031	Keyword FOR was expected
5032	Too many cursors declared
5033	Cursor is not open
5034	Cursor not declared
5035	Cursor is already open
5036	INTO list does not match SELECT list
5037	Specified cursor is not updateable
5038	Keyword TO expected
5039	No CURRENT record
5040	Exclusive tables cannot be locked
5041	Keyword SHARE or EXCLUSIVE expected
5042	Keyword MODE expected
5043	Keyword IN expected
5044	Keyword SELECT expected
5045	Keyword JOIN expected
5046	Invalid join expression
5047	Operation not supported on a VIEW
5048	View not declared

5049	VIEW already declared
5050	View name was expected
5051	Column '<column name>' not found
5052	Table privilege expected
5053	User name was expected
5054	User name <user name> was not found

Accessing RMS Data Files

On OpenVMS, Recital Terminal Developer and the Recital Database and Mirage Servers support access to the following fixed length RMS File types:

- RMS Sequential
- RMS Indexed Sequential
- RMS Relative

Data access is achieved through an RMS Bridge. This requires the creation of a Bridge file and an empty Recital table that has the same structure as the RMS file.

Creating the Recital Table

Create a Recital table with the same structure as the RMS file. The fields/columns in the structure file must exactly match the data type and length of those in the RMS file. The Recital table will have one byte more in total record length due to the Recital record deletion marker.

To create the table, use the SQL CREATE TABLE command or the Recital Terminal Developer for OpenVMS CREATE worksurface. The table should be given a '.str' file extension (rather than the default '.dbf') to signify that this is a structure file only.

Please see the end of this document for information on accessing VAX COBOL data types.

Creating the Bridge File

In Recital Terminal Developer for OpenVMS, the Bridge File can be created using the CREATE BRIDGE worksurface. For Recital Database and Mirage Server clients, the Bridge File can be created in two ways: by using an 'ini' file, or by the SQL CREATE BRIDGE command.

Maximums Widths

The following maximum widths apply to the bridge elements:

Element	Maximum Width in Characters	Description
Type	10	Bridge type: RMSSEQ, RMSIDX, RMSREL
External	80	External file name
Metadata	80	Recital 'structure' table name
Alias	10	Alias name
Index	50	Index key or filename

CREATE BRIDGE (SQL)

The CREATE BRIDGE SQL command defines and creates the bridge in one step:

e.g.

```
exec sql
CREATE BRIDGE rmsseqdemo.dbf
TYPE "RMSSEQ"
EXTERNAL "rmsseq.dat"
METADATA "rmsseqdemo.str"
ALIAS "rmsseqdemo";
```

or

```
exec sql
CREATE BRIDGE rmsseqdemo.dbf
AS "type=RMSSEQ;external=rmsseq.dat;metadata=rmsseqdemo.str;alias=rmsseqdemo";
```

For RMS Indexed Sequential files, the RMS index keys to be used can also be included in the bridge definition. Up to 7 different keys may be specified:

e.g.

```
exec sql
CREATE BRIDGE rmsidxdemo.dbf
TYPE "RMSIDX"
EXTERNAL "rmsidx.dat"
METADATA "rmsidxdemo.str"
ALIAS "rmsidxdemo"
INDEX "acc_prefix+acc_no,acc_prefix+str(ord_total)";
```

or

```
exec sql
CREATE BRIDGE rmsidxdemo.dbf
AS "type=RMSIDX;external=rmsidx.dat;metadata=rmsidxdemo.str;alias=rmsidxdemo;;
indexkey1=acc_prefix+acc_no;indexkey2=acc_prefix+str(ord_total)";
```

For RMS Sequential and RMS Relative files, up to 7 Recital single indexes can be built and associated with the bridge.

e.g.

```
exec sql
CREATE BRIDGE rmsreldemo.dbf
TYPE "RMSREL"
EXTERNAL "rmsrel.dat"
METADATA "rmsreldemo.str"
ALIAS "rmsreldemo"
INDEX "ind1.ndx,ind2.ndx,ind3.ndx";
```

or

```
exec sql
CREATE BRIDGE rmsreldemo.dbf
AS "type=RMSREL;external=rmsrel.dat;metadata=rmsreldemo.str;alias=rmsreldemo;;
indexkey1=ind1.ndx;indexkey2=ind2.ndx,indexkey3=ind3.ndx";
```


CREATE BRIDGE FROM <ini>

Firstly, an 'ini' file should be created on the server in the data directory where the external data file is held. The ini file has the following contents:

```
[bridge]
bridgetype=<bridgetype>
externalname=<name of the external data file>
databasename=<name of the Recital structure table>
alias=<the name to use to access your file>
indexkey1=<optional RMS index key or Recital index filename>
indexkey2=<optional RMS index key or Recital index filename>
indexkey3=<optional RMS index key or Recital index filename>
indexkey4=<optional RMS index key or Recital index filename>
indexkey5=<optional RMS index key or Recital index filename>
indexkey6=<optional RMS index key or Recital index filename>
indexkey7=<optional RMS index key or Recital index filename>
```

e.g. rmsreldemo.ini

```
[bridge]
bridgetype=RMSREL
externalname=rmsrel.dat
databasename=rmsreldemo.str
alias= rmsreldemo
indexkey1=ind1.ndx
indexkey2=ind2.ndx
```

e.g. rmsidxdemo.ini

```
[bridge]
bridgetype=RMSIDX
externalname=rmsidx.dat
databasename=rmsdemo.str
alias=rmsidxdemo
indexkey1=acc_prefix+acc_no
indexkey2=acc_prefix
```

NOTE: Recital Terminal Developer users can use the MODIFY BRIDGE to add in details of newly built Recital indexes. In client/server environments the SQL CREATE BRIDGE or 4GL CREATE BRIDGE FROM <ini> command needs to be reissued.

Then the CREATE BRIDGE command should be issued:

```
create bridge rmsidxdemo.dbf from rmsidxdemo
```

Using the Bridge

The Bridge can now be used. To access the RMS file, use the 'alias' specified in the Bridge definition.

e.g.

```
Select * from rmsseqdemo
```

Accessing VAX COBOL Data Types

The following table provides details of the COBOL data types that can be directly accessed by RECITAL using the RECITAL RMS Bridge.

COBOL Picture Clause	COBOL Usage Clause	RECITAL Data type	Storage in bytes
PIC 9(n)[n <=18]	USAGE IS DISPLAY	(N)umeric	n
PIC 9(n)[n <=18]	USAGE IS COMP-3	(P)acked	Variable
PIC 9(n)[n <=4]	USAGE IS COMP	(S)hort	2
PIC 9(n)[5 <=n <=9]	USAGE IS COMP	(I)nteger	4
PIC 9(n)[10 <=n <=18]	USAGE IS COMP	(Q)uad	8
PIC S9(n)[n <=4]	USAGE IS COMP	(S)hort	2
PIC S9(n)[5 <=n <=9]	USAGE IS COMP	(I)nteger	4
PIC S9(n)[10 <=n <=18]	USAGE IS COMP	(Q)uad	8
PIC S9(n)[10 <=n <=18]	USAGE IS INDEX	(I)nteger	4
PIC S9(n)[10 <=n <=18]	USAGE IS POINTER	(I)nteger	4
PIC S9(n)[10 <=n <=18]	USAGE IS COMP-1	(R)eal	4
PIC S9(n)[10 <=n <=18]	USAGE IS COMP-2	(F)loat	8
PIC S9(n)[n <=18]	USAGE IS COMP-3	(P)acked	Variable
PIC 9(n)[n <=18]	USAGE IS COMP-3	(P)acked	Variable
PIC X(n)[n <=254]	USAGE IS DISPLAY	(C)haracter	n
PIC A(n)[n <=254]	USAGE IS DISPLAY	(C)haracter	n
PIC 9(n)V9(s)	USAGE IS DISPLAY	(S)hort	2
PIC S9(n)V9(s)[(n+s) <=4]	USAGE IS COMP	(S)hort	2
PIC S9(n)V9(s)[5<=(n+s)<=9]	USAGE IS COMP	(I)nteger	4
PIC S9(n)V9(s)[10<=(n+s)<=18]	USAGE IS COMP	(Q)uad	8
PIC 9(n)V9(s)[n <=18]	USAGE IS COMP-3	(P)acked	Variable
PIC S9(n)V9(s)[n <=18]	USAGE IS COMP-3	(P)acked	Variable
PIC S9(n)[n <=18]	USAGE IS DISPLAY	not supported	
PIC S9(n)[n <=18]	USAGE IS DISPLAY SIGN IS TRAILING	not supported	
PIC S9(n)[n <=18]	USAGE IS DISPLAY SIGN IS LEADING	not supported	
PIC S9(n)[n <=18]	USAGE IS DISPLAY SIGN IS TRAILING SEPARATE	not supported	
PIC S9(n)[n <=18]	USAGE IS DISPLAY SIGN IS LEADING SEPARATE	not supported	
PIC S9(n)V9(s)[(n+s) <=18]	USAGE IS DISPLAY SIGN IS TRAILING	not supported	
PIC S9(n)V9(s)[(n+s) <=18]	USAGE IS DISPLAY SIGN IS TRAILING	not supported	
PIC S9(n)V9(s)[(n+s) <=18]	USAGE IS DISPLAY SIGN IS TRAILING SEPARATE	not supported	
PIC S9(n)V9(s)[(n+s) <=18]	USAGE IS DISPLAY SIGN IS LEADING SEPARATE	not supported	

NOTE:

The storage occupied packed decimal data types is calculated as follows:

if (n+s) is odd then storage = ((n+s)+1)/2
else storage = ((n+s)+2)/2

When defining the “width” for binary data types, this value denotes the output display width. The storage occupied by the data type is as specified above.

When defining the number of decimal places for binary data types, this value represents the “scale” of the value. When the field is referenced, RECITAL scales it down by successive divisions of 10, as specified by “scale”, and evaluates all arithmetic in double precision floating point. When fields of this type are updated, then the result to be stored in the field is again re-scaled.

Accessing C-ISAM Data Files

On UNIX and Linux platforms, Recital Terminal Developer, Recital Mirage Server and the Recital Database Server support access to Informix compliant C-ISAM files.

Data access is achieved through a C-ISAM Bridge. This requires the creation of a Bridge file and an empty Recital table that has the same structure as the C-ISAM file.

Creating the Recital Table

Create a Recital table with the same structure as the C-ISAM file. The fields/columns in the structure file must exactly match the data type and length of those in the C-ISAM file. The Recital table will have one byte more in total record length due to the Recital record deletion marker.

To create the table, use the SQL CREATE TABLE command or the Recital Terminal Developer CREATE worksurface. The table should be given a '.str' file extension (rather than the default '.dbf') to signify that this is a structure file only.

Please see the end of this section for information on matching Informix and Recital data types.

Creating the Bridge File

In Recital Terminal Developer, the Bridge File can be created using the CREATE BRIDGE worksurface. For Recital Database and Mirage Server clients, the Bridge File can be created in two ways: by using an 'ini' file, or by the SQL CREATE BRIDGE command.

Maximums Widths

The following maximum widths apply to the bridge elements:

Element	Maximum Width in Characters	Description
Type	10	Bridge type: CISAM
External	80	External file name
Metadata	80	Recital 'structure' table name
Alias	10	Alias name

'ini' file

Firstly, an 'ini' file should be created on the server in the data directory where the C-ISAM file is held. The ini file has the following structure:

```
[bridge]
bridgetype=<bridgetype>
externalname=<name of the C-ISAM file>
databasename=<name of the Recital structure table>
alias=<the name to use to access your file>
```

e.g. cisamdemo.ini

```
[bridge]
bridgetype=CISAM
externalname=cisam.dat
databasename=cisamstru.str
alias=cisamdemo
```

NOTE: There should be no white space either side of the '=' signs.

The Bridge file can now be created from the ini file. This can be given a '.dbf' file extension (rather than the default '.brg') so that it can be accessed like a normal Recital table. The SQL EXECUTE IMMEDIATE command is used to run the Recital/4GL CREATE BRIDGE FROM command:

e.g.

```
create bridge cisamdemo.dbf from cisamdemo
```

CREATE BRIDGE (SQL)

The CREATE BRIDGE SQL command defines and creates the bridge in one step:

e.g.

```
exec sql
CREATE BRIDGE cisamdemo.dbf
TYPE "CISAM"
EXTERNAL "cisamdemo.dat"
METADATA "cisamdemo.str"
ALIAS "cisamdemo";
```

//or

```
exec sql
CREATE BRIDGE cisamdemo.dbf
AS "TYPE=CISAM;EXTERNAL=cisamdemo.dat;METADATA=cisamdemo.str;ALIAS=cisamdemo";
```

Using the Bridge

The Bridge can now be used. To access the C-ISAM file, use the 'alias' specified in the Bridge definition.

e.g.

```
Select * from cisamdemo
```

Data Types

Informix	Recital
Byte	Numeric
Char	Character
Character	Character
Date	Date
Datetime	Character
Decimal	Numeric
Double Precision	Float
Float	Real
16 Bit Integer	Short
Integer	Numeric
Interval	Character
32 Bit Long	Integer
Money	Numeric
Numeric	Numeric
Real	Numeric
Smallfloat	Numeric
Smallint	Numeric
Text	Unsupported
Varchar	Character

C-ISAM RDD Error Messages

The following errors relate to the use of the Recital CISAM Replaceable Database Driver (RDD). They can be received as an 'errno <expN>' on Recital error messages:

ERRNO()	Error Description
100	Duplicate record
101	File not open
102	Invalid argument
103	Invalid key description
104	Out of file descriptors
105	Invalid ISAM file format
106	Exclusive lock required
107	Record claimed by another user
108	Key already exists
109	Primary key may not be used
110	Beginning or end of file reached
111	No match was found
112	There is no "current" established
113	Entire file locked by another user
114	File name too long
115	Cannot create lock file
116	Memory allocation request failed
117	Bad custom collating
118	Duplicate primary key allowed
119	Invalid transaction identifier
120	Exclusively locked in a transaction
121	Internal error in journaling
122	Object not locked

Using Xbase Data Files

To facilitate the migration of Xbase applications, Recital products contain the following features:

REPLACEABLE DATABASE DRIVERS (SET FILETYPE)

Replaceable Database Drivers allow Xbase tables and indexes to be used in their native format. The Replaceable Database Drivers are not available on OpenVMS or 64-bit UNIX. Please see below and the main SET FILETYPE entry in the SET COMMANDS documentation for further details.

DBCONVERT | CONVERT

The DBCONVERT utility and the CONVERT Recital/4GL command can be used to convert binary Xbase files to Recital format and to alter the line feed character of ASCII files to the correct format for the new host platform. The dbconvert utility is available in Recital Terminal Developer; the CONVERT command can also be used with Recital server products. For more on the DBCONVERT utility, please see below. The CONVERT command is in the main COMMANDS documentation.

SET COMPATIBLE

The SET COMPATIBLE command can be used in all Recital products to give added compatibility with Xbase dialects. For full details, please see the main entry in the SET COMMANDS documentation.

SET FILECASE

The SET FILECASE command determines whether references to file and directory names on UNIX and Linux are case-sensitive. If SET FILECASE is OFF, all file and directory names are treated as lower case. If SET FILECASE is ON, file and directory names are case-sensitive. For full details, please see the main entry in the SET COMMANDS documentation.

compat.db

The first time you start Recital Terminal Developer, or at any subsequent startup when the *compat.db* file does not exist in the Recital Terminal Developer home directory, you will be prompted to select the Language and Database compatibility settings. These correspond to the SET COMPATIBLE and SET FILETYPE settings and once saved are written to the *compat.db* file, which will be run every time Recital Terminal Developer is started. The following choices are available:

- Recital
- Visual FoxPro
- FoxPro
- FoxBase
- dBASE IV
- dBase 3
- Clipper 5
- Clipper '87

The settings dialog can be redisplayed by issuing the SET COMPATIBLE command with no ON, OFF or TO clause.

fox

Calling Recital Terminal Developer with the *fox* script will automatically SET FILETYPE TO FOXPRO and SET COMPATIBLE TO FOXPRO. This can also be achieved by passing the *-f* argument to the *db* script.

Replaceable Database Drivers

Recital provides Replaceable Database Drivers (RDDs) for popular Xbase products. These allow transparent use and creation of data and index files in the following product formats:

dBASE
Clipper
FoxPro

Use the SET FILETYPE command to set the required filetype.

SET FILETYPE TO [RECITAL | DBASE3 | DBASE4 | DB4 | FOXBASE | FOXPRO | FOXPLUS |
CLIPPER | VFP]

Example

```
use demo
select accounts
set filetype to foxpro
copy to foxacc
dir
```

The file type is shown in parentheses next to the file name in the DIR listing. Notice that although we have specified that the file type should be FOXPRO, the file foxacc.dbf has (DB3) next to it: where the file has no memo fields, as in this case, all the file types use the dBASE III+ standard format.

Example

```
select customer
copy to foxcust
dir
```

The customer table does have a memo field and so, therefore, does the copy. The file foxcust.dbf is shown with (FP2) next to its name.

The Replaceable Database Drivers can be used to enable a once only transfer of data into Recital format when data and applications are being migrated from Xbase products. Each table is opened, then copied to Recital format by ensuring that SET FILETYPE is set to Recital. They can also be used on a continuous basis where the same data tables and indexes need to be used by both Recital products and one of the Xbase products.

One further use of the Replaceable Database Drivers is where you need to transfer data files between binary incompatible Recital platforms, since the formats are the same across all platforms.

DBCONVERT

The DBCONVERT utility can be used to convert Xbase files into Recital format. The utility is run from the Operating System prompt, with the following syntax:

```
$ dbconvert <filetype> [<filename>]
```

Xbase binary files must be converted into Recital format before they can be used, unless they are being accessed using the Replaceable Database Drivers. ASCII format files can be used without conversion, but the conversion will correct the line feed format of the file to that of the new host platform. The following files can be converted:

File Contents	File Extension	Renamed extension (OpenVMS)	Renamed extension (UNIX/Linux)	Converted file extension
table	dbf	old_dbf	o_dbf	dbf
program	prg	old_prg	o_prg	prg
format	fmt	old_fmt	o_fmt	fmt
memory	mem	old_mem	o_mem	mem
report	frm	old_frm	o_frm	frm
report	frx	old_frx	o_frx	frm
text	txt	old_txt	o_txt	txt

The file types are as follows:

File Type	Description
ALL	All convertible files in the current directory
DBF	Tables and Memos
FMT	Screen Format files
FRX	FoxPro Report Format files
MEM	Memory files
PRG	Program files
TXT	Text files
FRM	Report Format files

The optional <filename> can be used to specify a single file or a group of files that match a pattern.

```
$ dbconvert dbf conv*.dbf
```

If the <filetype> is ALL, a 'config.db' file will automatically be created in the current directory containing the following compatibility commands:

```
set pcfilter on
set pcgraphics on
set prompt to "."
set pkeys on
set inkeydelay on
```

Xbase RDD Error Messages

The following list of error messages can be received when using native Xbase file formats via Recital.

Error	Description
-201	File write failure
-202	File read failure
-203	Memory allocation error
-204	File pointer reposition failed
-205	File not found
-206	File corrupted
-207	Bad user specified key expression
-208	No more file handles available
-209	No index pages loaded
-210	Index page was not loaded
-211	File close failure
-212	Invalid command
-213	Invalid file handle number
-214	Invalid filename
-215	Invalid date
-216	Invalid time
-217	File not in .DBT format
-218	Invalid Xbase file version
-219	File header length error
-220	Last file change date in error
-221	Invalid parameter address (NULL)
-222	Invalid index key type
-223	Invalid index key length
-224	Index key item length invalid
-225	Invalid index root page
-226	Invalid maximum number of index keys per page
-227	Invalid number of fields
-228	Invalid field name
-229	Invalid field length
-230	Invalid number of decimal places
-231	Invalid field data type
-232	Invalid record length
-233	Invalid data
-234	Invalid memo soft line length
-235	MDX flag in DBF file is invalid
-236	File open for read only
-237	File locking violation
-238	Sharing buffer overflow
-239	Path not found
-240	Access to file denied
-241	Invalid access code
-242	File must be locked first
-243	Destination device has changed
-244	Invalid minimum number of index keys per page
-245	Some files remain open
-246	Cannot open file
-247	Flush to disk failed

-248	Invalid index tag handle
-249	Invalid block size
-250	Invalid index tag name
-251	Invalid block adder size
-252	Invalid maximum number of index tags
-253	Invalid index tag table element length
-254	Invalid index tag count
-255	Unknown index key format switches
-256	Unknown switch error
-257	Index tag already open
-258	Windows GlobalLock() failed
-259	Windows task lookup failed
-260	Internal lock buffer overflow
-261	Internal lock buffer underflow

ASCII CHART

Hexadecimal	Decimal	Screen	Control Sequence	Control Key
00	0		NUL	CTRL-@
01	1		SOH	CTRL-A
02	2		STX	CTRL-B
03	3		ETX	CTRL-C
04	4		EOT	CTRL-D
05	5		ENQ	CTRL-E
06	6		ACK	CTRL-F
07	7		BEL	CTRL-G
08	8		BS	CTRL-H
09	9		HT	CTRL-I
0A	10		LF	CTRL-J
0B	11		VT	CTRL-K
0C	12		FF	CTRL-L
0D	13		CR	CTRL-M
0E	14		SO	CTRL-N
0F	15		SI	CTRL-O
10	16		DLE	CTRL-P
11	17		DC1	CTRL-Q
12	18		DC2	CTRL-R
13	19		DC3	CTRL-S
14	20		DC4	CTRL-T
15	21		NAK	CTRL-U
16	22		SYN	CTRL-V
17	23		ETB	CTRL-W
18	24		CAN	CTRL-X
19	25		EM	CTRL-Y
1A	26		SUB	CTRL-Z
1B	27		ESC	CTRL-[
1C	28		FS	CTRL-\
1D	29		GS	CTRL-]
1E	30		RS	CTRL-^
1F	31		US	CTRL-_
20	32			
21	33	!		
22	34	“		
23	35	#		
24	36	\$		
25	37	%		
26	38	&		
27	39	‘		
28	40	(
29	41)		
2A	42	*		
2B	43	+		
2C	44	,		
2D	45	-		
2E	46	.		
2F	47	/		
30	48	0		

31	49	1		
32	50	2		
33	51	3		
34	52	4		
35	53	5		
36	54	6		
37	55	7		
38	56	8		
39	57	9		
3A	58	:		
3B	59	;		
3C	60	<		
3D	61	=		
3E	62	>		
3F	63	?		
40	64	@		
41	65	A		
42	66	B		
43	67	C		
44	68	D		
45	69	E		
46	70	F		
47	71	G		
48	72	H		
49	73	I		
4A	74	J		
4B	75	K		
4C	76	L		
4D	77	M		
4E	78	N		
4F	79	O		
50	80	P		
51	81	Q		
52	82	R		
53	83	S		
54	84	T		
55	85	U		
56	86	V		
57	87	W		
58	88	X		
59	89	Y		
5A	90	Z		
5B	91	[
5C	92	\		
5D	93]		
5E	94	^		
5F	95	_		
60	96	`		
61	97	a		
62	98	b		
63	99	c		
64	100	d		
65	101	e		

66	102	f		
67	103	g		
68	104	h		
69	105	i		
6A	106	j		
6B	107	k		
6C	108	l		
6D	109	m		
6E	110	n		
6F	111	o		
70	112	p		
71	113	q		
72	114	r		
73	115	s		
74	116	t		
75	117	u		
76	118	v		
77	119	w		
78	120	x		
79	121	y		
7A	122	z		
7B	123	(
7C	124			
7D	125)		
7E	126	~		

Upgrading From Pre-9.0 Versions

Database tables (.dbf files) used by the Recital 9.0 and later product lines use a different file structure to previous Recital versions.

Therefore, before you can use your existing Recital data tables, they must be converted to the new file structure. This can be done by issuing the following command from the operating system prompt:

```
$ dbconvert ver90 <table name>.dbf
```

or, to convert all dbf files in the current directory:

```
$ dbconvert ver90
```

Recital Corporation strongly recommends that you perform a full backup of your Recital applications upgrading and converting your tables. The dbconvert utility will attempt to backup your existing data tables prior to converting, so you will need sufficient disk space available to hold these files. The Table (.dbf) file, the Memo (.dbt) file and the Index (.dbx) file will be backed up (to a <basename>.o_dbf file, a <basename>.o_dbt file and a <basename>.o_dbx file respectively). The Data Dictionary (.dbd) is not backed up as it remains unchanged. Production tag index files are recreated by the dbconvert process, but single index files (.ndx) will need to be rebuilt manually.

Upgrading From Pre-8.0 Versions

As part of Recital Corporation's Year 2000 Certification Program, the Recital 8.0 release was enhanced to provide date field storage up to the Year 2270. Because of this, the database tables (.dbf files) used by the Recital 8.0 and later product lines use a different file structure to previous Recital versions.

Therefore, before you can use your existing Recital data tables, they must be converted to the new file structure. This can be done by issuing the following command from the operating system prompt:

```
$ dbconvert year2000 <table name>.dbf
```

or, to convert all dbf files in the current directory:

```
$ dbconvert year2000
```

Recital Corporation strongly recommends that you perform a full backup of your Recital applications upgrading and converting your tables. The dbconvert utility will attempt to backup your existing data tables prior to converting, so you will need sufficient disk space available to hold these files. Note that only the .dbf file will be backed up (to a <table name>.v73 file), as the Data Dictionary (.dbd) and Memo (.dbt) files remain unchanged.

Upgrading dbx Indexes From Pre-8.3 Versions

The permissible path and filename length stored in a .dbx multiple index file was increased in Recital 8.3. All multiple index files (.dbx files) must be deleted and rebuilt when upgrading from pre-Recital 8.3 versions.

Example

```
erase file company.dbx
use company nodbx
index on co_code tag code
index on lower(co_name) tag name
```

NOTE: Recital Corporation recommends that program files are recompiled and indexes rebuilt whenever a new version of the software is installed.

Optimizing Indexes with SYNCNUM

See Also

CONVERT, DB_INDEXSEQNO, SYNCNUM

Recital Character and date indexes can be optimized based on the use of the pseudo column SYNCNUM to ensure that all keys are unique. Every row/record in a table has a unique SYNCNUM value for that table.

The optimization is enabled using the DB_INDEXSEQNO environment variable/symbol. This updates the SYNCNUM column of all new rows added to tables. In turn, any new character or date indexes include the SYNCNUM column automatically: character indexes have SYNCNUM added to the end of the expression, date indexes are converted to DTOS() and have the SYNCNUM added to the end. Numeric index are not affected.

The SYNCNUM pseudo column for existing Recital 9 tables can be populated using the dbconvert utility and the CONVERT command. The dbconvert utility is used from the operating system prompt:

```
$ dbconvert index <table name>.dbf
```

or, to convert all dbf files in the current directory:

```
$ dbconvert index
```

For more information on the CONVERT command, please see the main CONVERT entry under Recital 4GL Commands.

When DB_INDEXSEQNO is set to true, any tables and indexes that have not previously been converted, will be converted to use the SYNCNUM optimization as they are opened. It is however recommended that you use “dbconvert index” or “convert index” to manually convert your tables before setting DB_INDEXSEQNO to true.

DB_CONFIG

Class

Environment Variables / Symbols

Purpose

Used to define the full path of the Recital Database Server and Recital Mirage Server configuration files

See Also

DB_MIRAGE

Description

The DB_CONFIG environment variable / symbol is set to the full path name of the configuration file for the Recital Database and Mirage Servers. By default this is */usr/recitalxx/UAS/config.db*.

Products

Recital Database Server, Recital Mirage Server

DB_DATADIR

Class

Environment Variables / Symbols

Purpose

Used to define a default data directory for SQL database creation and for the dbexec utility

See Also

CREATE DATABASE, CREATE TABLE, DROP DATABASE, DROP TABLE, USE, DBEXEC, SET PATH

Description

The DB_DATADIR environment variable / symbol is used to define a default data directory for SQL database creation and the dbexec utility. SQL Databases in Recital are implemented as directories containing files that correspond to the tables and associated files in the database. Operating System file protection can be applied individually to the files for added security. The directory is a sub-directory of the Recital data directory as set in DB_DATADIR. SQL databases are created using the SQL CREATE DATABASE command.

The dbexec utility is used to run program files via the Recital Database and Mirage Servers without connecting from a client. If DB_DATADIR is set to a directory, this directory will be added to the search path and is equivalent to using the SET PATH command.

Example

```
# DB_DATADIR="/usr/recital/data"; export DB_DATADIR
# dbexec myapp
```

```
// create_dat.prg
CREATE DATABASE hr;
USE hr;
// end of create_dat.prg
> ? getenv([DB_DATADIR])
/usr/recital/data
> do create_dat
> ? default()
/usr/recital/data/hr
```

Products

Recital Database Server, Recital Mirage Server

DB_DATE

Class

Environment Variables / Symbols

Purpose

Used to define the date format for the license expiry date

See Also

SET DATE

Description

The DB_DATE environment variable / symbol is used to define the date format for the license expiry date. DB_DATE is set to “american”. This setting must not be changed, as it could cause the license expiry date to be read incorrectly and an expired license error to be generated. To change the date format for use within the Recital environment, the SET DATE command should be used.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_ENCRYPTION

Class

Environment Variables / Symbols

Purpose

Used to determine whether the Recital Database and Mirage Servers expect username and password information in connection strings to be encrypted.

See Also

APPEND FROM, COPY FILE, COPY STRUCTURE, COPY TO, DECRYPT, DIR, ENCRYPT, USE, SET ENCRYPTION, SQL INSERT, SQL SELECT, SQL UPDATE

Description

The DB_ENCRYPTION environment variable / symbol is used to determine whether the Recital Database and Mirage Servers expect username and password information in connection strings to be encrypted. If DB_ENCRYPTION is set to true (or “yes” or “on”), username and password information in connection strings will be interpreted as DES3 encrypted data. If DB_ENCRYPTION is set to false (or “no” or “off”), such data is not treated as DES3 encrypted.

Clients must have the appropriate settings to send the username and password information in DES3 encrypted format.

Products

Recital Database Server, Recital Mirage Server

DB_ERRORDIR

Class

Environment Variables / Symbols

Purpose

Used to specify the directory in which error files should be created

See Also

ERROR(), MESSAGE(), SET ERRORVERSION

Description

Set this environment variable (Linux/UNIX) or symbol (OpenVMS) to the name of the directory in which error files should be created. If this environment variable / symbol is not set, error files will be created in the directory that was current at the time of the error.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_FIRECATLOG

Class

Environment Variables / Symbols

Purpose

Used to enable or disable Firecat Web Server Common Log Format access logging

See Also

DB_LOGDIR, USERLOG()

Description

The DB_FIRECATLOG environment variable / symbol is used to enable or disable Firecat Web Server access logging. With DB_FIRECATLOG set to “True”, “On” or “Yes”, a Common Log Format access log called firecat.log is created and updated in the log directory indicated by DB_LOGDIR.

Products

Recital Web Developer

DB_FOXMEM

Class

Environment Variables / Symbols

Purpose

Used to specify the creation of FoxPro binary format '.mem' files for the SAVE TO command

See Also

RESTORE, SAVE, SET COMPATIBLE, SET FILETYPE, Using Xbase Files

Description

The DB_FOXMEM environment variable / symbol is used to specify that FoxPro binary format '.mem' files should be created by the SAVE TO command, rather than Recital ASCII format '.mem' files.

DB_FOXMEM should be set to "true", "yes" or "on" to enable this behavior.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_FOXPLUSBUGS

Class

Environment Variables / Symbols

Purpose

Used for increased SCO FoxPlus compatibility

See Also

SET COMPATIBLE, SET FILETYPE, Using Xbase Files

Description

The DB_FOXPLUSBUGS environment variable / symbol is used to increase SCO FoxPlus compatibility. If DB_FOXPLUSBUGS is set to “yes” or “on”, behavior of the sys(5), sys(2003) and sys(2004) functions will be compatible with SCO FoxPlus.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_FOXPROKEYS

Class

Environment Variables / Symbols

Purpose

Used for increased FoxPro READKEY() function compatibility

See Also

READKEY(), SET COMPATIBLE, SET FILETYPE, Using Xbase Files

Description

The DB_FOXPROKEYS environment variable / symbol is used to increase FoxPro compatibility. If DB_FOXPROKEYS is set to “yes” or “on”, the return value from READKEY() for the [RETURN] / [ENTER] key is 15.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_HOSTNAME

Class

Environment Variables / Symbols

Purpose

Used to define the Recital Server hostname/IP address when the server machine has more than one network card/IP address

See Also

DB_LICENSE_SERVER

Description

The DB_HOSTNAME environment variable is used to define the Recital Server hostname/IP address when the server machine has more than one network card/IP address.

DB_HOSTNAME should be set to the required hostname or IP address.

Products

Recital Database Server, Recital Mirage Server

DB_HTTP_ALLOW

Class

Environment Variables / Symbols

Purpose

Used to specify the name of a file containing IP addresses allowed to connect to the Recital Firecat Web Server

See Also

DB_HTTP_DENY, DB_UAS_ALLOW, DB_UAS_DENY

Description

The name of a file in the UAS directory containing IP addresses that are allowed to connect to the Recital Firecat Web Server. The default filename is the same as for the main Recital Server (defined in the DB_UAS_ALLOW environment variable), hosts.allow. IP addresses should be listed in the file one to a line.

Products

Recital Firecat Web Server

DB_HTTP_DENY

Class

Environment Variables / Symbols

Purpose

Used to specify the name of a file containing IP addresses denied from connecting to the Recital Firecat Web Server

See Also

DB_HTTP_ALLOW, DB_UAS_ALLOW, DB_UAS_DENY

Description

The name of a file in the UAS directory containing IP addresses that are not allowed to connect to the Recital Firecat Web Server. The default filename is the same as for the main Recital Server (defined in the DB_UAS_DENY environment variable), hosts.deny. IP addresses should be listed in the file one to a line.

Products

Recital Firecat Web Server

DB_INDEXSEQNO

Class

Environment Variables / Symbols

Purpose

Used for data and character index optimization

See Also

Upgrading from pre-9.0 versions, INDEX, USE

Description

The DB_INDEXSEQNO environment variable / symbol environment variable has been added to help with optimizing character and date indexes. The optimization is based on the use of the pseudo column SYNCNUM to ensure that all keys are unique. Every row/record in a table has a unique SYNCNUM value for that table.

When DB_INDEXSEQNO is set to true, any tables and indexes that have not previously been converted, will be converted to use the SYNCNUM optimization as they are opened. It is however recommended that you use “dbconvert index” or “convert index” to manually convert your tables before setting DB_INDEXSEQNO to true.

New rows added to tables will always have the SYNCNUM column updated with a new unique number regardless of the DB_INDEXSEQNO setting. Once .dbx files have been converted to use SYNCNUM optimization, the DB_INDEXSEQNO value has no effect.

To disable SYNCNUM optimization after it has been enabled, you must set the value of DB_INDEXSEQNO to false and then recreate all the index and tag files. Tag files will need to be recreated from scratch, so tables must be opened with USE <table> NODBX.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_LICENSE_SERVER

Class

Environment Variables / Symbols

Purpose

Used to define the Recital Server license hostname/IP address when the server machine has more than one network card/IP address

Description

The DB_LICENSE_SERVER environment variable is used to define the Recital Server license hostname/IP address when the server machine has more than one network card/IP address.

DB_LICENSE_SERVER should be set to the required hostname or IP address.

Products

Recital Database Server, Recital Mirage Server

DB_LOCAL_LOGIN

Class

Environment Variables / Symbols

Purpose

Used to determine whether the Recital Database and Mirage Servers allow local connections from specific non-authenticated users.

Description

The DB_LOCAL_LOGIN environment variable / symbol is used to determine whether the Recital Database and Mirage Servers allows local connections from specific non-authenticated users. Normally, checks are made on all users attempting to connect to the Server to ensure that they are valid users. If DB_LOCAL_LOGIN is set to true (or “on” or “yes”), then the username/password combinations of recital/recital and ?? are not checked providing the client is on the same machine as the Server. If DB_LOCAL_LOGIN is set to false (or “off” or “no”), then all connecting users must provide valid username and password information for the machine in question. DB_LOCAL_LOGIN has no effect on connections from remote clients; all remote connections must be authenticated.

Products

Recital Database Server, Recital Mirage Server

DB_LOGDIR

Class

Environment Variables / Symbols

Purpose

Used to specify the directory in which log files should be created

See Also

SET SYSLOGGING, DB_LOGVER

Description

Set this environment variable (Linux/UNIX) or symbol (OpenVMS) to the name of the directory in which log files should be created.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_LOGVER

Class

Environment Variables / Symbols

Purpose

Used to specify the directory whether multiple log files should be created

See Also

SET SYSLOGGING, DB_LOGDIR

Description

If this environment variable (Linux/UNIX) or symbol (OpenVMS) is set to “NO”, “no”, “OFF” or “off”, the Recital Database and Mirage Servers will overwrite existing log files rather than creating multiple numbered log files. By default, multiple numbered log files will be created when logging is on.

Products

Recital Database Server, Recital Mirage Server

DB_MAXROW

Class

Environment Variables / Symbols

Purpose

Used to define the maximum number screen rows for Recital Mirage

See Also

COL(), MAXCOL(), MAXROW(), ROW(), @...SAY

Description

The DB_MAXROW is a Recital Mirage Server environment variable / symbol used to define the maximum number of screen rows for Recital Mirage applications. By default the maximum number of screen rows is 25. DB_MAXROW can be set in the profile.uas configuration file (Linux, UNIX, OpenVMS) or in the HKEY_LOCAL_MACHINE\SOFTWARE\Recital\UAS\NetServer\DB_MAXROW Registry entry (Windows).

Please note: The SIZE parameter in the HTML file (browser based applications) or in the Mirage.conf file (executables) must also be changed to match DB_MAXROW. For browser based applications, the applet height may also need to be modified.

Products

Recital Mirage Server

DB_MAXWKA

Class

Environment Variables / Symbols

Purpose

Used to define the maximum number of available workareas

See Also

SELECT, USE, SELECT(), WORKAREA(), SET VIEW

Description

The DB_MAXWKA environment variable / symbol is used to define the maximum number of available workareas. By default this is set to 20. The maximum value for DB_MAXWKA is 256.

Please note: DB_MAXWKA should be kept to the minimum required to avoid unnecessary memory allocation.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_MIRAGE

Class

Environment Variables / Symbols

Purpose

Used to define the directory or folder containing the mirage_styles.conf configuration file

See Also

DB_CONFIG

Description

The DB_MIRAGE environment variable / symbol is set to the full path name of the directory or folder that contains the mirage_styles.conf configuration file. The path name should not contain a trailing directory separator character.

Products

Recital Mirage Server

DB_MIRAGE_COMMAND

Class

Environment Variables / Symbols

Purpose

Used to define a global command for Recital Mirage applications

See Also

DB_MIRAGE_CONFIG, DB_MIRAGE_DIRECTORY, DB_MIRAGE_PATH, DB_UAS_ALLOW, DB_UAS_DENY

Description

DB_MIRAGE_COMMAND is used to define the command to be used for Recital Mirage applications. This allows the value of the *command* applet parameter to be overridden. The ability to override the applet parameters contained in the HTML file centrally from the server protects the server from misuse of these parameters.

Products

Recital Mirage Server

DB_MIRAGE_CONFIG

Class

Environment Variables / Symbols

Purpose

Used to define a global configuration file for Recital Mirage applications

See Also

DB_MIRAGE_COMMAND, DB_MIRAGE_DIRECTORY, DB_MIRAGE_PATH, DB_UAS_ALLOW, DB_UAS_DENY

Description

DB_MIRAGE_CONFIG is used to define the configuration file to be used for Recital Mirage applications. This allows the value of the *config* applet parameter to be overridden. The ability to override the applet parameters contained in the HTML file centrally from the server protects the server from misuse of these parameters.

Products

Recital Mirage Server

DB_MIRAGE_DIRECTORY

Class

Environment Variables / Symbols

Purpose

Used to define a global login directory for Recital Mirage applications

See Also

DB_MIRAGE_COMMAND, DB_MIRAGE_CONFIG, DB_MIRAGE_PATH, DB_UAS_ALLOW, DB_UAS_DENY

Description

DB_MIRAGE_DIRECTORY is used to define the login directory to be used for Recital Mirage applications. This allows the value of the *directory* applet parameter to be overridden. The ability to override the applet parameters contained in the HTML file centrally from the server protects the server from misuse of these parameters.

Products

Recital Mirage Server

DB_MIRAGE_PATH

Class

Environment Variables / Symbols

Purpose

Used to define a global file search path for Recital Mirage applications

See Also

DB_MIRAGE_COMMAND, DB_MIRAGE_CONFIG, DB_MIRAGE_DIRECTORY,
DB_UAS_ALLOW, DB_UAS_DENY

Description

DB_MIRAGE_PATH is used to define the search path to be used for Recital Mirage applications. This allows the value of the *path* applet parameter to be overridden. The ability to override the applet parameters contained in the HTML file centrally from the server protects the server from misuse of these parameters.

Products

Recital Mirage Server

DB_NOPAM

Class

Environment Variables / Symbols

Purpose

Used to disable PAM (Pluggable Authentication Module) user authentication

Description

The DB_NOPAM environment variable is used to disable PAM (Pluggable Authentication Module) user authentication for the Recital Database and Mirage Servers. If DB_NOPAM is set to “on” or “yes”, PAM is disabled and users are authenticated using FTP. If DB_NOPAM is unset or is set to “off” or “no”, PAM will be used for Server user authentication on those systems on which it is supported.

DB_NOPAM should be set to TRUE in the UAS/profile.xml file when using the Universal ODBC Driver for Linux/UNIX.

Products

Recital Database Server, Recital Mirage Server (Both UNIX/Linux only)

DB_ODBC_INI

Class

Environment Variables / Symbols

Purpose

Used to define the odbc.ini file for Linux/UNIX ODBC data source definitions

Description

The DB_ODBC_INI environment variable is used to define the odbc.ini file for Linux/UNIX ODBC data source definitions.

It is set in the UAS/profile.xml file for the Universal ODBC Driver for Linux/UNIX.

Products

Recital Database Server (UNIX/Linux only)

DB_OPTLOG

Class

Environment Variables / Symbols

Purpose

Used to determine whether the evaluation of logical expressions should be optimized

See Also

SET OPTLOG

Description

The DB_OPTLOG environment variable / symbol is used to determine whether the evaluation of logical expressions should be optimized. If DB_OPTLOG is set to “no” (or “NO”), evaluating a logical expression causes the entire expression to be evaluated. If DB_OPTLOG is unset, or is set to a value other than “no” or “NO”, the evaluation of the logical expression is optimized. The optimization causes the evaluation to stop as soon as the result of the evaluation is known.

For example:

If DB_OPTLOG is set to “no”, the following command will cause an error, assuming no function called ‘crash’ exists.

```
➤ ? .F. and crash()
```

If DB_OPTLOG is unset, or set to a non-‘no’ value, no error will occur, since the evaluation process will stop after the .F. has been evaluated. At this point, the result of the expression can only be .F. (false).

In post-8.2 versions, DB_OPTLOG also affects the optimization of logical expressions containing the OR operator.

For example:

If DB_OPTLOG is set to “no”, the following command will cause an error, assuming no function called ‘crash’ exists.

```
➤ ? .T. or crash()
```

If DB_OPTLOG is unset, or set to a non-‘no’ value, no error will occur, since the evaluation process will stop after the .T. has been evaluated. At this point, the result of the expression can only be .T. (true).

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_PRINTEREJECT

Class

Environment Variables / Symbols

Purpose

Used to determine whether the printer eject command is issued by SET PRINT OFF

See Also

SET PRINT, SET PRINTER

Description

The DB_PRINTEREJECT environment variable / symbol is used to determine whether the printer eject command is issued by SET PRINT OFF. If DB_PRINTEREJECT is set to “on” or “yes”, the printer eject command will be issued automatically by SET PRINT OFF. If DB_PRINTEREJECT is set to “off” or “no”, the printer eject command will not be issued automatically by SET PRINT OFF.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_PRINTERROR

Class

Environment Variables / Symbols

Purpose

Used to define check for and report errors in Linux/UNIX printing scripts

See Also

SET PRINTER

Description

The DB_PRINTERROR environment variable is used to determine whether checking should be carried out on Linux/UNIX printing scripts. If DB_PRINTERROR is set to “on” or “yes”, Recital will check for errors in Linux/UNIX printing scripts and report them.

The SET PRINTER TO \\SPOOLER command is used to redirect printer output to a system printer. This picks up the environment variable DB_PRINT, which by default is set to the OS script file print.unix in the UD directory. This file can be replaced or modified to suit the particular printing environment.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer (All UNIX/Linux only)

DB_PROCDIR

Class

Environment Variables / Symbols

Purpose

Used to define a default procedures and programs directory for the dbexec utility

See Also

DB_DATADIR, DBEXEC, SET PATH

Description

The DB_PROCDIR environment variable / symbol is used to define a default procedures and programs directory for the dbexec utility. The dbexec utility is used to run program files via the Recital Database and Recital Mirage Servers without connecting from a client. If DB_PROCDIR is set to a directory, this directory will be added to the search path and is equivalent to using the SET PATH command.

Example

```
# DB_PROCDIR="/usr/recital/UAS/netprocs"; export DB_PROCDIR
# dbexec myapp
```

Products

Recital Database Server, Recital Mirage Server

DB_REREAD_COLORFILE

Class

Environment Variables / Symbols

Purpose

Used to determine which default.col file is used by SET COLOR TO

See Also

SET COLOR

Description

The DB_REREAD_COLORFILE environment variable / symbol is used to determine which default.col file is used by SET COLOR TO when the current directory contains a different default.col file from the starting directory. The default.col file in the current directory will be used when a SET COLOR TO is issued unless DB_REREAD_COLORFILE="no". If DB_REREAD_COLORFILE="no" then SET COLOR TO will restore the colors defined at startup.

Products

Recital Mirage Server, Recital Terminal Developer

DB_RUNLOG

Class

Environment Variables / Symbols

Purpose

Used to determine whether calls to operating system commands should be enabled or disabled.

See Also

!, !!, RUN, RUN(), SET RUNCLEAR, SET RUNWAIT

Description

The DB_RUNLOG environment variable / symbol is used to determine whether calls to operating system commands should be enabled or disabled. This is particularly useful when debugging applications, particularly when they are being migrated from one product to another or from one platform to another. If DB_RUNLOG is true (or "yes" or "on"), calls to operating system commands are disabled and any such calls are logged in the recital.log file. If DB_RUNLOG is false (or "no" or "off"), the calls are processed as normal.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_RUNOPTS

Class

Environment Variables / Symbols

Purpose

Used to issue an stty command after leaving Recital to execute a RUN command.

See Also

!, !!, RUN, RUN(), SET RUNCLEAR, SET RUNWAIT, DB_RUNLOG, DB_RUNOPTS2

Description

The DB_RUNOPTS environment variable is used to issue an stty command after leaving Recital Terminal Developer to execute a RUN command. It works in conjunction with DB_RUNOPTS2, which can issue an stty command to reset the environment before returning to Recital Terminal Developer. For example, they can be used to set echo on and back off again for a RUN command:

```
# profile.db extract
DB_RUNOPTS="stty echo icrnl"      ;export DB_RUNOPTS
DB_RUNOPTS2="stty -echo -icrnl"   ;export DB_RUNOPTS2
```

Products

Recital Terminal Developer

DB_RUNOPTS2

Class

Environment Variables / Symbols

Purpose

Used to issue an stty command before returning to Recital after executing a RUN command.

See Also

!, !!, RUN, RUN(), SET RUNCLEAR, SET RUNWAIT, DB_RUNLOG, DB_RUNOPTS

Description

The DB_RUNOPTS2 environment variable is used to issue an stty command before returning to Recital Terminal Developer after executing a RUN command. It works in conjunction with DB_RUNOPTS, which can issue an stty command to set the environment after leaving Recital Terminal Developer. For example, they can be used to set echo on and back off again for a RUN command:

```
# profile.db extract
DB_RUNOPTS="stty echo icrnl"      ;export DB_RUNOPTS
DB_RUNOPTS2="stty -echo -icrnl"   ;export DB_RUNOPTS2
```

Products

Recital Terminal Developer

DB_SAMBA

Class

Environment Variables / Symbols

Purpose

Used to enable support of Samba locking

See Also

SET COMPATIBLE, SET FILETYPE

Description

The DB_SAMBA environment variable / symbol is used to enable support of Samba locking.

For more information on configuring Recital products to work with Samba, please see http://www.recital.com/products_samba.htm.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer (All UNIX/Linux only)

DB_SAVEHISTORY

Class

Environment Variables / Symbols

Purpose

Used to determine whether persistent history will be written to the command.his file

See Also

Error Handling & Debugging, SET DOHISTORY, SET HISTORY

Description

The DB_SAVEHISTORY environment variable / symbol is used to determine whether persistent history will be written to the command.his command history file. If DB_SAVEHISTORY is true, a persistent command history will be written to the command.his file in the current working directory. This command history is not reset or restarted by a new session, but persists across multiple sessions. Previous commands logged to the command.his file can be recalled using the cursor keys at the command line.

If DB_SAVEHISTORY is false, writing and reading of the command.his file is disabled.

Products

Recital Terminal Developer

DB_TMPDIR

Class

Environment Variables / Symbols

Purpose

Used to specify the full path name of a directory or folder to be used by the SYS(3) function for temporary file creation

See Also

SET TMPDIR, SYS(3)

Description

The DB_TMPDIR environment variable / symbol is used to define a directory or folder that will be used by the SYS(3) function for temporary file creation. If DB_TMPDIR is not defined, SYS(3) returns a unique file name without path information. Defining DB_TMPDIR causes this directory to be included in the return value from SYS(3). Please note that the DB_TMPDIR should include a trailing directory separator.

Example

```
? getenv([DB_TMPDIR])
```

```
/usr/tmp/
```

```
? sys(3)
```

```
/usr/tmp/000cf30006
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_TSINDEX

Class

Environment Variables / Symbols

Purpose

Must be set to enable Text Search Index functionality

See Also

TSPOS(), TSWORD(), SET TSLENGTH

Description

The DB_TSINDEX environment variable / symbol is used to enable Text Search Index functionality. It must be set to “ON” for Text Search Indexes to operate correctly.

Text Search Indexes are created using the TSWORD() function in the key and searched using the TSPOS() function.

Example

```
? getenv([DB_TSINDEX])
```

ON

use example

index on tsword(first_name+last_name,1) to namesearch

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_UAS_ALLOW

Class

Environment Variables / Symbols

Purpose

Used to specify the name of a file containing IP addresses allowed to connect to the Recital Database or Recital Mirage Servers

See Also

DB_UAS_DENY

Description

The name of a file in the *UAS* directory containing IP addresses that are allowed to connect to the Recital Database or Recital Mirage Servers. The default filename is *hosts.allow*. IP addresses should be listed in the file one to a line.

Products

Recital Database Server, Recital Mirage Server

DB_UAS_DENY

Class

Environment Variables / Symbols

Purpose

Used to specify the name of a file containing IP addresses denied from connecting to the Recital Database or Recital Mirage Servers

See Also

DB_UAS_ALLOW

Description

The name of a file in the *UAS* directory containing IP addresses that are not allowed to connect to the Recital Database or Recital Mirage Servers. The default filename is *hosts.deny*. IP addresses should be listed in the file one to a line.

Products

Recital Database Server, Recital Mirage Server

DB_UNIXPATH

Class

Environment Variables / Symbols

Purpose

Used to determine whether the Linux/UNIX PATH should be included in the SET PATH list

See Also

SET PATH

Description

The DB_UNIXPATH environment variable / symbol is used to determine whether the directories specified in the Linux/UNIX PATH environment setting should be automatically included in the SET PATH list. If DB_UNIXPATH is set to “on” or “yes”, the directories are automatically included. If DB_UNIXPATH is set to “off” or “no”, the directories are not included.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer (All UNIX/Linux only)

DB_USERLOG

Class

Environment Variables / Symbols

Purpose

Used to specify a directory and filename to which user specific logging information should be written

See Also

USERLOG()

Description

The DB_USERLOG environment variable / symbol is used to specify a directory and filename to which user specific logging information should be written. With DB_USERLOG set to a valid filename, the USERLOG() function can be used to write to the log for debugging or audit trail purposes.

By default, the DB_USERLOG environment variable is:

```
DB_USERLOG="${DB_ROOT}log/${LOGNAME}.log"
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DB_XMLREP

Class

Environment Variables / Symbols

Purpose

Used to automatically switch on XML transaction journaling

See Also

XMLCOUNT(), XMLCREATEDTD(), XMLFIRST(), XMLNEXT(), XMLVALIDATE()

Description

The DB_XMLREP environment variable / symbol is used to automatically switch on XML transaction journaling.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer