

JAXBase

Programming Guide



Jon Lee Walker
JLWalker.Dev@gmail.com
Copyright 2025 ©

The Team

Design

Jon Lee Walker

Documentation

Jon Lee Walker

Development

Jon Lee Walker

Testers

If I have seen further, it is by standing on the shoulders of giants -- Isaac Newton

Acknowledgements

C. Wayne Ratliff & JPL
Ashton Tate

Dr. Dave Fulton, Bill Ferguson, and the FoxBase/FoxPro Teams
Nantucket Corporation
Borland dBase Team
Microsoft Visual FoxPro Team
StackOverflow.Com
Learn.Microsoft.Com

Contents

Forward.....	26
JAXBase Features	27
JAXBase Departures from XBase	28
Current Status	29
Release Goals	30
Features Not Yet Supported	31
Licensing, etc.	32
About this Book	Error! Bookmark not defined.
Command Reference.....	33
?	34
??	35
& (macro substitution).....	36
&&, *, NOTE	37
ADD	38
Add Class	38
Add Table	40
ALTER TABLE	42
APARAMETERS.....	44
APPEND.....	45
Append	45
Append Blank.....	46
Append From Array	47
Append From.....	49
APPPEND FROM JSON	52
ASSERT	53
AVERAGE	54
BEGIN TRANSACTION	56
BLANK	57
BROWSE	58

BUILD	60
BUILD APPLICATION	60
BUILD EXECUTABLE.....	61
BUILD PROJECT.....	62
CALCULATE	64
CANCEL.....	66
DO CASE ... CASE... OTHERWISE...ENDCASE	67
CD	68
CLEAR	69
CLEAR.....	69
CLEAR CLASS.....	71
CLEAR CONSOLE	72
CLEAR EVENTS	74
CLEAR ERRORS	75
CLEAR MACROS.....	76
CLEAR MEMORY.....	77
CLEAR PROGRAM.....	78
CLEAR TYPEAHEAD	79
CLOSE	80
CLOSE ALL	80
CLOSE ALTERNATE	81
CLOSE DATABASE.....	82
CLOSE DATASESSION.....	83
CLOSE INDEXES.....	84
CLOSE TABLES	85
COMPILE	86
CONTINUE.....	87
COPY FILE.....	88
COPY INDEXES	89
COPY STRUCTURE	90

COPY TO ARRAY	91
COPY TO	93
COUNT	97
CREATE TABLE / CURSOR.....	99
CREATE DATABASE.....	101
CREATE FORM	102
CREATE LABEL	103
CREATE MENU	104
CREATE PROJECT.....	105
CREATE QUERY.....	106
CREATE REPORT	107
DEBUG	108
DEBUGOUT.....	109
DEFINE CLASS	110
DELETE	111
DELETE FILE.....	112
DIMENSION	113
DIRECTORY / DIR.....	115
DISPLAY DATABASE	116
DISPLAY FILES	117
DISPLAY MEMORY.....	118
DISPLAY OBJECTS	120
DISPLAY STATUS	122
DISPLAY STRUCTURE	124
DISPLAY TABLES	125
DO	127
DO Program.....	127
DO FORM	129
DO WHILE/ENDDO	131
DO CASE/CASE/OTHERWISE/ENDCASE	133

DO/UNTIL.....	135
DOEVENTS.....	137
DROP TABLE	138
EDIT.....	139
ERROR.....	141
EXIT	142
EXTERNAL.....	143
FOR/ENDFOR	144
FOREACH/ENDFOR	146
FUNCTION.....	147
GATHER	148
GETEXPR	150
GOTO.....	151
HELP.....	153
IF/ELSEIF/ELSE/ENDIF.....	154
IMPORT.....	156
INDEX	157
INSERT.....	159
KEYBOARD.....	161
LIST.....	162
LOCATE.....	163
LOCAL	165
LOOP.....	166
LPARAMETERS	167
LPROCEDURE.....	169
MD.....	170
MODIFY	171
Modify Class	171
Modify Command.....	172
Modify File.....	173

Modify Form	174
Modify Label.....	175
Modify Menu.....	176
Modify Project	177
Modify Query.....	178
Modify Report.....	179
Modify Structure.....	180
MOUSE	181
ON.....	182
ON ESCAPE	182
ON KEY LABEL	183
ON SHUTDOWN.....	184
OPEN.....	185
PACK	187
PARAMETERS	188
PLAY	191
PRIVATE	192
PROCEDURE.....	193
PUBLIC	194
QUIT	195
RD	196
READ EVENTS	197
RECALL.....	198
REINDEX.....	199
RELEASE	200
RELEASE Command	200
RELEASE CLASSLIB	201
RELEASE PROCEDURE	202
REMOVE	203
REMOVE CLASS.....	203

REMOVE TABLE	204
RENAME	205
RENAME Command.....	205
RENAME CLASS.....	206
Rename Table.....	207
REPLACE	208
REPLACE Command.....	208
REPLACE FROM ARRAY.....	210
RESTORE	212
RESTORE FROM.....	212
RESTORE MACROS.....	214
RESUME.....	215
RETRY	216
RETURN	217
SAVE	218
SAVE CLASS Command.....	218
SAVE MACROS	219
SAVE TO	220
SCAN / ENDSCAN	221
SCATTER	223
SEEK Command	225
SELECT Command.....	227
SET Commands	229
SET ALTERNATE Command	230
SET ASSERTS Command	231
SET AUTOINCERROR Command	232
SET AUTOSAVE Command	233
SET BELL Command	234
SET BLOCKSIZE Command	235
SET BROWSEIME Command	236

SET CARRY Command	237
SET CENTURY Command.....	238
SET CLASSLIB Command	239
SET COLLATE Command	240
SET CONFIRM Command	241
SET CONSOLE Command.....	242
SET COVERAGE Command.....	244
SET CPCCOMPILE Command.....	245
SET CPDIALOG Command	246
SET CURRENCY Command.....	247
SET CURSOR Command	248
SET DATABASE Command.....	249
SET DATASESSION Command.....	250
SET DATE Command.....	251
SET DEBUGOUT Command.....	252
SET DECIMALS Command	253
SET DEFAULT Command	254
SET DELETED Command	255
SET DEVELOPMENT Command.....	256
SET ESCAPE Command	257
SET EVENTLIST Command	258
SET EVENTTRACKING Command	259
SET EXACT Command.....	260
SET EXCLUSIVE Command	261
SET FDOW Command.....	262
SET FIELDS Command.....	263
SET FILTER Command.....	264
SET FIXED Command.....	265
SET FULLPATH Command.....	266
SET FWEEK Command.....	267

SET HEADINGS Command.....	268
SET HELP Command	269
SET HOURS Command	270
SET INDEX Command.....	271
SET LOCK Command.....	272
SET MACKEY Command.....	273
SET MEMOWIDTH Command.....	274
SET MESSAGE Command	275
SET MULTILOCKS Command.....	276
SET NAMING Command	277
SET NEAR Command	278
SET NOCPTRANS Command.....	279
SET NOTIFY Command	280
SET NULL Command	281
SET NULLDISPLAY Command	282
SET ODOMETER Command	283
SET ORDER Command	284
SET PATH Command.....	285
SET POINT Command.....	286
SET PROCEDURE Command	287
SET REFRESH Command	288
SET RELATION Command	289
SET REPROCESS Command	290
SET RESOURCE Command	291
SET SAFETY Command	292
SET SECONDS Command.....	293
SET SECURITY Command	294
SET SEPARATOR Command	295
SET SKIP Command.....	296
SET SKIP OF Command.....	297

SET SPACE Command	298
SET SQLCONNECTION Command	299
SET STEP Command	300
SET STRICTDATE Command	301
SET SYSMENU Command	302
SET TABLEPROMPT Command	303
SET TABLEVALIDATE Command	304
SET TALK Command	305
SET TEXTMERGE Command	306
SET TEXTMERGE DELIMITERS Command	307
SET TOPIC Command	308
SET TOPIC ID Command	309
SET TRBETWEEN Command	310
SET TYPEAHEAD Command	311
SKIP Command.....	312
SORT	313
STORE.....	315
SUM.....	316
SUSPEND	318
TEXT / ENDTEXT.....	319
THROW.....	321
TRY/CATCH/FINALLY/ENDTRY.....	322
UNLOCK	324
UPDATE	325
UPDATE FROM	326
USE	327
WAIT	329
WITH/ENDWITH	330
ZAP	331
Function Reference	333

ABS()	334
ACLASS()	335
ACOPY()	336
ACOS()	338
ADATABASES()	339
ADBOBJECTS()	340
ADDBS()	342
ADDPROPERTY()	343
ADEL()	344
ADIR()	346
AELEMENT()	348
AERROR()	350
AEVENTS()	352
AFIELDS()	353
AFONT()	355
AINS()	356
AINSTANCE()	358
ALEN()	359
ALIAS()	360
ALINES()	361
ALLTRIM()	363
ALOCFILE()	365
AMEMBERS()	367
APRINTOBJECTS()	369
ASC()	370
ASCAN()	371
ASELOBJ() – Under consideration.	373
ASESSIONS()	374
ASIN()	375
ASORT()	376

ASTACKINFO()	378
ASUBSCRIPT()	380
AT()	382
AT_C()	384
ATC()	385
ATCC()	387
ATCLINE()	388
ATLINE()	389
ATN2()	392
ATOKENS()	393
AUSED()	395
AWORKAREAS()	397
BETWEEN()	398
BINDEVENT()	399
BINTOC()	400
BITAND()	402
BITCLEAR()	403
BITLSHIFT()	404
BITNOT()	405
BITOR()	406
BITRSHIFT()	407
BITSET()	408
BITTEST()	409
BITXOR()	410
BOF()	411
CANDIDATE()	412
CAPSLOCK()	413
CDOW()	414
CEILING()	415
CHR()	416

CHRTRAN()	417
CHRTRANC()	418
CMONTH()	419
COMMIT()	420
COMPILE()	421
COS()	422
CPCCONVERT()	423
CPCURRENT()	424
CPDBF()	425
CREATEBINARY()	426
CREATEOBJECT()	427
CTOBIN()	428
CTOD()	430
CTOT()	431
CURSORGETPROP()	432
CURSORSETPROP()	433
CURSORTOJSON()	434
CURSORTOXML()	435
CURVAL()	436
DATE()	438
DATETIME()	439
DAY()	441
DBC()	442
DBF()	443
DBGETPROP()	444
DBSETPROP()	445
DBUSED()	446
DELETED()	447
DESCENDING()	448
DIFFERENCE()	450

DIRECTORY()	451
DISKSPACE()	453
DYM()	454
DODEFAULT()	455
DOW()	456
DRIVETYPE()	458
DTOC()	459
DTOR()	460
DTOS()	461
DTOT()	462
EMPTY()	463
EOF()	465
ERROR()	466
EVALUATE()	467
EVL()	468
EXECSCRIPT()	469
EXP()	470
FCOUNT()	471
FDATE()	472
FIELD()	473
FILE()	474
FILETOSTR()	475
FILTER()	476
FLOCK()	477
FLOOR()	479
FONTMETRIC()	480
FOR()	481
FORCEEXT()	482
FORCEPATH()	483
FOUND()	484

FSIZE()	485
FTIME()	487
FULLPATH()	488
FV()	489
GETAUTOINCVALUE()	490
GETTCP()	491
GETDATE()	492
GETDIR()	493
GETENV()	495
GETFILE()	496
GETFIELDSTATE()	498
GETFONT()	499
GETNEXTMODIFIED()	500
GETJSON() Function	501
GETPICT()	503
GETPRINTER()	504
GETTIME()	505
GETWORDCOUNT()	506
GETWORDNUM()	507
GETCURSORADAPTER()	508
GOMONTH()	509
GUID()	510
HEADER()	511
HEXToint()	512
HOUR()	513
ICASE()	514
IDXCOLLATE()	515
IIF()	516
INDEXSEEK()	517
INKEY()	518

INLIST()	520
INPUTBOX()	521
INSMODE()	523
INT()	524
INTTOHEX()	525
ISALPHA()	526
ISBLANK()	527
ISDIGIT()	529
ISEXCLUSIVE()	530
ISFLOCKED()	531
ISLEADBYTE()	532
ISLOWER()	533
ISNULL()	534
ISODD()	535
ISPEN()	536
ISREADONLY()	537
ISRLOCKED()	538
ISUPPER()	539
JSONTOCURSOR()	540
JSONTOOBJ()	541
JUSTDRIVE()	542
JUSTTEXT()	543
JUSTFNAME()	544
JUSTFULLPATH()	545
JUSTPATH()	546
JUSTSTEM()	547
KEY()	548
LASTKEY()	549
LEFT()	550
LEFTC()	551

LEN()	552
LENC()	553
LIKE()	554
LIKEC()	555
LINENO()	556
LOADPICTURE()	557
LOCK()	558
LOG()	560
LOG10()	561
LOWER()	562
LTRIM()	563
LUPDATE()	564
MAX()	565
MDY()	566
MEMLINES()	567
MEMORY()	568
MESSAGE()	569
MESSAGEBOX()	570
MIN()	572
MINUTE()	573
MLINE()	574
MOD()	575
MONTH()	576
NDX()	577
NORMALIZE()	578
NUMLOCK()	579
NVL()	580
OBJTOCLIENT()	581
OBJTOJSON()	583
OCCURS()	584

OLDVAL()	585
ON()	586
ORDER()	588
OS()	590
PADC() / PADL() / PADR()	591
PARAMETERS()	592
PAYMENT()	593
PEMSTATUS()	594
PI()	595
PROGRAM()	596
PROPER()	597
PUTJSON()	598
PUTFILE()	600
PV()	601
QUARTER()	602
RAND()	603
RAT()	604
RATC()	606
RATLINE()	607
RECCOUNT()	609
RECNO()	610
RECSIZE()	611
RELATION()	612
REMOVEPROPERTY()	613
REPLICATE()	614
REQUERY()	615
RGB()	616
RIGHT()	617
RIGHTC()	618
RLOCK()	619

ROLLBACK()	621
ROUND()	622
RTOD()	623
RTRIM()	624
SAVEPICTURE()	625
SEC()	626
SECONDS()	627
SEEK()	628
SELECT()	630
SET()	631
SETFLDSTATE()	635
SIGN()	636
SIN()	637
SOUNDEX()	638
SPACE()	639
SQRT()	640
STR()	641
STRCONV()	642
STREXTRACT()	643
STRFORMAT()	645
STRTOFILE()	648
STRTRAN()	649
STUFF()	651
STUFFC()	653
SUBSTR()	654
SUBSTRC()	655
TAN()	656
TARGET()	657
TEXTMERGE()	658
TIME()	659

TRANSFORM()	660
TRIM()	661
TTOC()	662
TTOD()	664
TOSEC()	665
TXNLEVEL()	666
TXTWIDTH()	667
TYPE()	669
UNBINDEVENTS()	671
UPDATED()	672
UPPER()	673
USED()	674
VAL()	675
VARTYPE()	676
VERSION()	678
WEEK()	679
XMLCURSOR()	680
YEAR()	681
Notes	682
JAXBase Environment Objects and Variables	684
JAXBase Environment Variables	685
_JAXFolder	685
_JAXHome	685
_JAXTemp	685
JAXBase Environment Objects and Arrays	686
_JAX	687
_CPU	688
_DRIVES	689
_KEYBOARD	690
_MONITORS	691

_MOUSE.....	692
_PRINTERS	693
_REGEX	694
JAXBase Class Reference.....	697
_APPEND	698
_BROWSE	699
_EDIT	702
_ERROR	703
BARCODE	705
CHECKBOX.....	707
COLLECTION	708
COMBOBOX.....	709
COMMANDBUTTON	710
COMMANDGROUP	712
CONTAINER	713
CUSTOM	714
EDITBOX	716
EMPTY	718
FILE	719
FORM	720
FORMSET.....	723
FTP	724
GRID	725
HTTP	726
IMAGE.....	727
LABEL	728
LINE.....	730
LISTBOX.....	731
MENU	732
OPTIONBUTTON	733

OPTIONGROUP.....	734
PAGE	735
PAGEFRAME.....	736
PIPE.....	737
POP3	738
PRINTER	739
SHAPE	740
SMS	741
SMTP	742
SOUND	743
SPINNER.....	745
SQL	746
TEXTBOX.....	748
TCP.....	750
TIMER	751
UDP.....	752
VIDEO	753
JAXBase Concepts.....	756
Arrays, Variables, and Objects	757
Data Sessions and Work Areas	758
DBF Fields	759
Dates	760
Debugging and Error Handling	761
Using Indexes.....	762
Using the Keyboard Commands	763
Merging Text	764
Moving Data To and From a SQL Database	765
Code Page Support	766
Reading and Writing Barcodes.....	767
Tables and Cursors.....	768

Using One or More a SQL Database	769
Using the Regular Expression Object	770
Appendixes.....	772
Appendix A – File Types.....	773
Appendix B – Table File Structure	774
32-bit Table Structure	774
32-bit Table Field Sub-Records	775
64-bit Table Structure	776
64-bit Table Field Sub-Records	777
Appendix C – JAXBase Field Types.....	779
APPENDIX D – JAXBase Data Types	780
Appendix E – System Variables	781
Appendix F – JAXBase Data Sessions	782
Appendix G – DEF Files	783
Appendix H – Collation Sequences	784
JAXBase FAQ	786

Forward

This is a work in progress with a long timeline.

I have had this project on my mind for the better part of last 30 years and as I approach retirement, I decided it was time to start.

This is version 0.1 and is simply a partial document release. Version 0.4 will be the first binary release where the user will have a basic IDE that allows them to compile a few different types of files (such as PRG and SCX) with the ability to do simple input/output and work with tables and indexes.

I have been asked why I am doing this. The quick answer is because I love the XBase language and detest the corporate decision to abandon it because it gave too much power to the developer and not enough to the corporate bottom line.

Further, it is my opinion that, when done right, the XBase language is one of the most powerful languages ever developed for the desktop and is well within reach for a beginner to learn.

Additionally, computers are about handling data, and the XBase language is all about handling data in clear and concise ways.

I was tempted to make an open-source version of Visual FoxPro, but I decided that I wanted to improve the language by eliminating the "last century thinking" and improve the language constructs that remain. Additionally, there are desperately needed features that I think a modern XBase dialect should have. If you are knowledgeable in DBase or FoxPro, you will be quite familiar with JAXBase. You will also see a broad array of new features that allow you to do things not easily done in other dialects.

I don't know if anyone will be greatly interested in reviving the XBase language, but finding the answer to that question is my way of giving back to an industry that has given me so much.

XBase. It has been very, very good to me.

Jon Lee Walker
jwalker.dev@gmail.com

JAXBase Features

The JAXBase Project is an attempt to create a modern version of the venerable XBase language. As stated earlier, Version 0.4 will be the proof of concept and everything before Version 2.0 will be written in C# and .Net which is not the ideal platform for a cross-platform computer language.

Version 1.0 is will new features over legacy xBase dialects, including:

JSON support. Examples are SCATTER, GATHER, GETJSON(), JSONTOOBJ(), OBJTOJSON(), and PUTJSON() functions.

APARAMTERS captures and accepts all non-array parameters sent and stuffs them into a one-dimensional array.

New classes, including SQL, BARCODE, PRINTER, REPORT, LABEL, HTTP, and TCP.

Better backend SQL handling using the SQL class and JAXBase commands like:

```
CLOSE ALL  
CLEAR ALL  
  
* Open the TEST database and create the MyDB SQL object variable  
OPEN DATABASE TO MyDB USING "server=127.0.0.1 uid=root pwd=12345 database=test"  
  
* Since there isn't an open table in this work area, JAXBase assumes  
* you are accessing a table in the current SQL database.  
* MyDB!MyTable explicitly tells JAXBase to go get a SQL Table.  
SELECT * FROM MyCursor TO CURSOR SqlResults  
  
* Insert or append a record to the local cursor.  
* Extra code may be needed, depending on the type of primary key.  
INSERT INTO MyCursor (FirstName, LastName, BirthDate) VALUES ("JOHN", "SMITH", {2001/07/15})  
  
* Update the SQL Table using the cursor in this work area  
MyDB.UPDATE("MyTable")  
  
IF MyDB.AError[1].ErrorNo>0  
    ? "Error!"  
ENDIF  
  
CLOSE DATABASE MyDB
```

JAXBase Departures from XBase

JAXBase is highly object oriented and most legacy/MS-DOS commands and last-century paradigms have been removed.

The @ commands, color pairs, and color schemes no longer exist.

ON ERROR is gone. Use TRY/CATCH/FINALLY or the Error method in classes.

Menus and its related components are now a class instead of commands.

You can use most JAXBase table commands to interact with a SQL database and its tables. Additionally, the SQL class provides you with rich features, allowing you to use the specific syntax for your preferred SQL engine.

JAXBase is designed to be a true cross platform language. Windows will be the first targeted operating system with Linux support coming in Version 1. Operating system specific features that are not part of the language, but there will be ways to communicate with the operating system using external add-ons that communicate through one or more JAXBase communication classes.

When the XBase language was introduced, micro-computers had one floppy disk capable of holding under three hundred Kbytes which meant that every byte was precious and thus abbreviated commands were allowed. Today, clarity is much more important and JAXBase does not allow abbreviated commands except for a very few instances that are ubiquitous in the industry.

Proper spacing with commands is required. Expressions fewer requirements for proper spacing since A=B.OR.C=D is easily parsed by a computer.

There is no local database container support. Further, only simple indexes (IDX) are available.

There is no report or label preview rendering. Reports and labels are rendered using open-source office suites.

Current Status

2025-08-31 – Version 0.2

Version 0.1 is a partial document release to GitHub which can be considered a "Request for Comment" period. Version 0.2 will be a proposed Final Draft, and Version 0.3 will be the Final Draft for Version 1.0 of JAXBase.

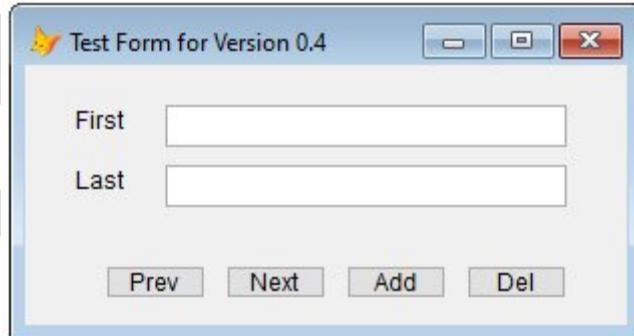
If you see a Status header, then that feature is not expected to be in the version 0.4 release.

The first public binary release will be Version 0.4 for Windows. At present, JAXBase is written in C# and is expected to remain in C# through Version 1.

Form, TextBox, Label, and CommandButton are the only visual classes slated for Version 0.4 of JAXBase. Most visual classes will be available in Version 0.6, while the rest are expected to show up by the Version 1.0 release.

The Version 0.4 binary will be released when a simple form containing labels, text boxes, and buttons can be correctly displayed, and a user can interact with the form and save data to a table.

This is the form that is being used to test JAXBase Version 0.4, as displayed by Visual FoxPro.



The code for table and index handling is written, though it is still in testing. Tables are currently set for exclusive access only. Version 0.8 will introduce table/record locking and sharing.

The compiler correctly tokenizes over 80% of the commands while about 20% of the runtime code is written. GitHub will be updated periodically with progress notes and documentation updates.

Release Goals

Version 1.0

- Most editors and tools will be written in JAXBase code with full source in the TOOLS folder.
- Simplistic report and label designer.
- DBF header, and record locking with multi-user support on tables.
- Support for MS SQL Server and MySQL databases.

Version 1.1

- Error correction release

Version 1.2

- Finalize and complete available classes for Version 1.

Version 1.3

- Completion of commands and functions slated for Version 1.

Version 2.0

- Move JAXBase to C++ or Rust and use a GUI framework like Qt.
- Enhanced report and label designers.
- Add support for PostgreSQL, Oracle, and other data engine back ends.
- Integration with IIS and Apache Web Server.
- Double-byte characters and advanced multi-Lingual support.

Features Not Yet Supported

There are features that will not be available prior to Version 2.

Double byte expressions and fields. Only UTF-8 strings are currently supported.

Code pages are not supported. A code page value is accepted by JAXBase commands, but it has no current meaning.

Only MACHINE collation is currently supported. Version 2 will correct that.

Report and Label support are not available before Version 1.0 and will then be limited but will improve with each subsequent release. The full report and label ecosystem is Slated for Version 2.

Direct multi-lingual support. I am working on ideas to make it easier to develop multi-lingual applications. These ideas are in the JAXBase: Muti-Lingual Applications document and input is always welcome.

Licensing, etc.

For now, the JAXBase system is Copyrighted, but it is expected to be released under the GNU General Public License version 2 (GPL-2.0) with the first source release.

JAXBase is written in C# for Windows using .NET, and efforts by others willing to help make JAXBase run on Linux will be very welcome. Version 2.0 is expected to move to C++ or Rust.

Security Concerns

JAXBase is under rapid development and there is no intention or expectation of meeting modern security standards. Development and integration of security will begin after the Version 1.0 release, and for that reason, JAXBase 1.x will be considered an educational tool.

Version 2.0 must be built for modern security requirements from the ground up.

Mission of the JAXBase Project

The mission of the JAXBase Project is to encourage use of open-source software and the renewal of the XBase language as a powerful and modern solution, allowing the thousands of legacy XBase applications to once again be relevant to Windows and Linux users.

Our Goals

To create and encourage the use and development of secure, open-source software.

To create an ecosystem that encourages the development of multilingual support.

To create a cross-platform language that frees the developer and user from hardware/OS concerns, licensing fees, and use restrictions.

To allow users to create responsive applications that can run on something as small as the amazing Raspberry PI and have no limits on the hardware to which it can be scaled.

To encourage further development of the XBase ecosystem, supporting the use of popular back-end SQL database engines, including ones that are not open-source.

To encourage integration with other open-source projects.

Command Reference

Please be aware of the important differences between JAXBase and most XBase language dialects.

- 1) The popularity of the XBase language started with DBase 2, back when you had a few hundred kilobytes of storage on a floppy disk. Since every byte was considered precious, most commands could be abbreviated to four characters. For instance, GOTO BOTTOM could be written GO BOTT. Modern systems offer multi-trillion-byte storage media and application source can span dozens or even hundreds of megabytes of code, making clarity and readability vital. Thus, the decision was made to not allow command abbreviations in the JAXBase language except for a very few instances which are mentioned in this reference, such as DIRECTORY / DIR.
- 2) A particular corporation made it difficult to write free text unless you send it to a console single window or write your own GUI library. The JAXBase free text commands ?, ??, DISPLAY, DIRECTORY, along with others will only display to the text console window which is normally hidden unless you SET CONSOLE ON.
- 3) JAXBase is very object oriented. In the typical XBase dialect, you reference reports, printers, and other objects through commands designed for those features and devices. JAXBase dispensed with that paradigm and instead uses classes to create reports and address devices.
- 4) Because JAXBase is designed to be cross-platform, it does not directly support OLE and other Windows specific technologies that are not available in Linux. However, OS specific access will be available through classes that can communicate with add-on components to support the parent operating system.

?

```
? [eExpression1] [, eExpression2] [, eExpression3]...
```

Remarks

Sends a carriage return followed by the comma delimited expression list to the console window, if open. If SET ALTERNATE or SET PRINT is turned on, then it will be sent to either or both of those two devices. If the console window is not open and ALTERNATE and PRINT is set off, the information is lost.

Example

```
A=4  
B=7  
? "Hello"  
? A,B,B+A
```

Note

The ? command first sends a text line terminator (Windows characters 0x0A and 0x0D, Linux character 0x0D) before displaying any expressions. The expressions are separated by commas, and each comma adds a space to the displayed text.

If SET ALTERNATE or SET PRINT is turned on, then it will be sent to either or both of those two devices. If the console window is not open and ALTERNATE and PRINT is set off, the information is lost.

??

```
?? [eExpression1] [, eExpression2] [, eExpression3]...
```

Remarks

Sends the comma delimited expression list to the console window, if open.

Example

```
A=4  
B=7  
? "Hello"  
?? A,B,B+A
```

Note

Unlike the ? command, the ?? command begins to display expressions at the current cursor location. The expressions are separated by commas, and each comma adds a space to the displayed text.

If SET ALTERNATE or SET PRINT is turned on, then it will be sent to either or both of those two devices. If the console window is not open, and ALTERNATE and PRINT are set off, the information is lost.

& (macro substitution)

&var

Status

Allowed in expressions in Version 0.4 and full support is slated for Version 1.0

Remarks

Macro substitution comes in two forms and is much more powerful in JAXBase than some popular xBase varieties. The two types are & and () macros.

& Macro

This is the standard replacement where there is a & and variable name pair (Example &a1) and the value inside the variable is inserted into the code and then processed. An example would be:

```
* Demonstrate macro substitution
STORE "B" to A
B=1
C=&A+2 && C contains 3
```

When JAXBASE encounters a command with a & macro, it solves the value of the macro, compiles the line of code, and then checks for any other & macros in that line of code. This means you can have a macro variable that contains another macro variable, and it will continue to solve for that line of code until there are no more & macros. If the macro variable contains multiple lines of code, the lines of code are resolved, and each line is checked for macros as they execute.

() Macro

This is another type of macro substitution and is faster but has limits. It can only be placed into the code where a literal is expected. These macros cannot be put into expressions or contain code for execution as they are simply meant to allow an expression to be placed where a literal is expected.

```
* Demonstrate literal substitution
B=1
A="Test.dbf"
SELECT (B)
USE (A)
```

Note

A ¯o with multiple lines of code is executed outside of the current program or form and is considered a separate PRG. Thus, the code in that macro must be self-contained as it only has access to variables of a global or private scope In essence, you are running a separate PRG.

A multiline macro cannot be sent parameters or RETURN a value unless executed using the EXECSCRIPT() function.

See Also

EXECSCRIPT() Function

&&, *, NOTE

```
&& [programming information]
* [programming information]
NOTE [programming information]
```

Remarks

The &&, *, and NOTE commands allow you to insert programming comments into the code, making the code easier to understand, or to give information that you want to place into the code. The comment information is stripped out of the compiled code.

While the * comment command must be the only command on the line, the && and // commands can appear anywhere, but at that point, the remainder of the statement becomes part of the comment and discarded during compilation.

Example

```
* HelloWorld.prg demonstration program
SET CONSOLE ON    && open the command window
? "Hello world"
```

ADD

Add Class

```
ADD CLASS ClassName [OF ClassLibraryName1] to ClassLibraryName2 [OVERWRITE]
```

Status

Slated for Version 1.0

Remarks

Copy a class from one class library to another.

Parameters

ClassName

Name of class to be added to the VCX file.

OF ClassLibraryName1

If included, copies the class from the supplied VCX file.

TO ClassLibraryName2

VCX file name that will have the class written to it.

OVERWRITE

If included, it will overwrite any matching class name.

Note

Use ADD CLASS to add a class definition to an existing class library or copy from one class library to another. A class can only be added to a VCX file.

ADD CLASS will work at runtime or in the IDE, but the VCX needs to be compiled before it can be used.

If the class name already exists in ClassLibraryName1 and OVERWRITE is not specified, an error is generated.

If you omit OF ClassLibraryName1, JAXBase will look for ClassName in all open class libraries. If the ClassName is not found, an error is generated.

An error message will be generated if you attempt to write to a class library that already has a class with the same name.

See Also

ACLASS() Function
AMEMBERS() Function
CREATE CLASS Command
CREATE CLASSLIB Command
DEFINE CLASS Command
MODIFY CLASS Command
RELEASE CLASSLIB Command
REMOVE CLASS Command
RENAME CLASS Command
SAVE CLASS Command
SET CLASSLIB Command

DRAFT

Add Table

```
ADD TABLE TableName [AS SQLTableName] [TO cDBName] [OVERWRITE] [NODATA]
```

Status

Slated for Version 0.6

Remarks

Specifies the name of the DBF table to copy to an open SQL Database.

Parameters

TableName

Name of DBF table to copy.

AS SQLTableName

If included, allows the renaming of the DBF table, else the table's filename stem will be used.

TO cDBName

Specifies the database connection to use, otherwise the current database connection is used.

OVERWRITE

If included, allows the overwriting of a SQL table if it exists.

NODATA

If included, only the table structure is copied to the SQL database.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

* Assumes a SQL Connection exists to a backend database
OPEN DATABASE TEST USING FILETOSTR("SQLConnect.dsn")

CREATE TABLE TEST1 (PKey C(18), Name C(30), Balance N(10,2))

* Send the structure to the SQL Server
ADD TABLE (TEST1) OVERWRITE NODATA
```

Note

The JAXBase table fields will be converted to corresponding SQL tables. For instance, memo fields will be converted to VarChar(MAX) in SQL Server databases and TEXT for MySQL databases. This command is used to quickly move a DBF into a SQL table. For better control of the process, see the SQL class.

See Also

CLOSE Commands
CREATE DATABASE Command
DISPLAY TABLES Command
OPEN DATABASE Command
REMOVE TABLE Command
SELECT DATABASE Command
SELECT DATASESSION Command
INDBC() Function
SQL Class

ALTER TABLE

```
ALTER TABLE TableName  
ADD|ALTER COLUMN FieldName FieldType [(nFieldWidth [,nPrecision])]  
[NULL | NOT NULL] [AUTOINC [NEXTVALUE nNextValue [STEP nStepValue]]]
```

Status

ALTER for DBF tables is slated for Version 0.6

NULL and AUTOINC support is slated for Version 2.0

Remarks

Use ALTER TABLE to change a field in a DBF table.

Parameters

ADD COLUMN

Command used to add a column to a table.

ALTER COLUMN

Command used to alter a column in a table.

FieldName

Literal character name of field to add or alter.

FieldType

Literal character value indicating a JAXBase field type. See [DBF Fields](#)

nFieldWidth

Literal numeric value indicating width of the field.

nPrecision

Literal numeric value indicating of decimal points.

NULL | NOT NULL (Slated for Version 2, ignored for now)

AUTOINC, NEXTVALUE, STEP (Slated for Version 2, ignored for now)

Example

```
* Demonstrate ALTER TABLE
CLEAR ALL
CLOSE ALL
SET SAFETY OFF
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

CREATE TABLE Test1 (Name C(20))
USE

ALTER TABLE Test1 ALTER COLUMN Name C(30)
ALTER TABLE Test1 ADD COLUMN Age N(3)

USE Test1
DISPLAY STRUCTURE
```

Note

ALTER TABLE requires exclusive access to a table. If the table being altered is open, even with an alias, it will be closed.

If the table cannot be opened exclusively, an error is generated.

Use the SQL class EXEC method to alter a table in a SQL database.

See Also

- ADD TABLE Command
- AFIELDS() Function
- CREATE TABLE
- INDEX Command
- MODIFY STRUCTURE Command
- OPEN DATABASE Command
- SQL Class

APARAMETERS

APARAMETERS ArrayName

Remarks

APARAMETERS places all parameter values passed to the routine into a local array specified as aVar. If a variable exists with that name, it becomes unavailable as the new local variable takes precedence. If no parameters were passed, then aVar will have one element holding a .null. value.

The APARAMETERS statement must be the first executable command in a routine, otherwise an error will be generated.

By default, parameters sent using DO / WITH commands are handle where Arrays and Objects are passed by reference and all others are sent by value. However, the APARMETERS command will generate an error if you attempt to send an array as a parameter.

Parameters

ArrayName

Name of the array variable to place parameters passed from the calling program.

Example

```
* Demonstrate APARAMETERS
Do Testing1 WITH 1,3,9,"A"
RETURN

PROCEDURE Testing1
APARAMETERS aTest
LIST MEMO LIKE aTest
RETURN
```

Note

If APAMETERS is used in the main program, then any parameters passed to it from outside the program, such as from a command prompt, are placed into the array by value.

See Also

- ALEN()
- DIMENSION
- DO FORM
- DO program
- LPARAMETERS
- PARAMETERS
- User Defined Functions

APPEND

Append

APPEND

Status

Slated for after Version 1.0

Remarks

Displays the append screen allowing you to directly apply data to the end of the currently open table or cursor in the work area, if it is not read-only. If no table or workarea is open, it will generate an error.

Example

* Demonstrate APPEND

CLEAR ALL

CLOSE ALL

SET CONSOLE ON

ACTIVATE CONSOLE

CLEAR

CREATE TABLE TEST (Name C(30))

APPEND

? RECCOUNT()

USE

See Also

APPEND BLANK

APPEND FROM ARRAY Command

BROWSE Command

EDIT Command

IMPORT Command

INSERT command

SELECT – SQL Command

Append Blank

```
APPEND BLANK IN nWorkArea | cTableAlias]
```

Remarks

Appends a blank record to the end of the DBF table. If APPEND BLANK is applied to a SQL Table then an error is generated. The Append Window is not open with APPEND BLANK. You can alter the record with BROWSE, EDIT, or the REPLACE command.

Parameters

IN nWorkArea | cTableAlias

Causes ADD BLANK to create a record in a different work area or table

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

CREATE TABLE TEST1 (Name C(20))

SELECT 2
CREATE TABLE TEST2 (Name C(20))

SELECT 3
CREATE TABLE TEST3 (Name C(20))

* Add a blank record to TEST1
APPEND BLANK IN 1

* Add a blank record to TEST2
APPEND BLANK IN TEST2

CLOSE TABLES ALL
```

Note

If the table has a UNIQUE or CANDIDATE index open, you could generate an error if a blank key already exists.

See Also

APPEND

APPEND FROM ARRAY Command

BROWSE Command

EDIT Command

IMPORT Command

INSERT command

SELECT – SQL Command

Append From Array

```
APPEND FROM ARRAY ArrayName [FOR lExpression]
    [FIELDS FieldList | FIELDS LIKE Skeleton | FIELDS EXCEPT]
```

Status

Slated for Version 0.6

Remarks

Append elements from an array into a table.

Parameters

ArrayName

Name of array containing data for the table.

FOR lExpression

Conditional expression that controls whether a record is appended from the array.

FIELDS FieldList

Specifies the fields to populate from the array.

FIELDS LIKE

Specifies a field skeleton where matching fields are populated from the array.

FIELDS EXCEPT

Specifies fields to skip when populating from an array

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
DIMENSION aRecs[3]
aRecs[1] = "TOM"
aRecs[2] = "DICK"
aRecs[3] = "MARY"
```

```
CREATE TABLE TEST1 (Name C(20), Age I)
APPEND FROM ARRAY aRecs
```

Note

Memo, General, and Blob fields are ignored in APPEND FROM ARRAY

When a table is open for shared use, APPEND FROM ARRAY locks the table header while records are being added.

If a one-dimensional array has more elements than fields, the remainder are ignored.

If a two-dimensional array has more columns than fields being appended in each row, the remaining elements in each row of the array are ignored.

If there are not enough columns in an array to fill all fields, the rest are left blank.

See Also

APPEND

APPEND BLANK

BROWSE Command

EDIT Command

IMPORT Command

INSERT command

SELECT – SQL Command

DRAFT

Append From

Appending from DBF or Cursor

```
APPEND FROM cFile [FIELDS FieldList] [FOR lExpression]
[AS nCodePage]
```

Appending from Plain Delimited Text File

```
APPEND FROM cFile [FIELDS FieldList] [FOR lExpression] [AS nCodePage]
[DELIMITED [WITH Delimiter | WITH BLANK | WITH TAB | WITH CHARACTER Delimiter]]
[AS nCodePage]
```

Appending from File of TYPE

```
APPEND FROM cFile [FIELDS FieldList] [FOR lExpression]
[TYPE CSV| SDF | [XLS | XLSX | ODS [SHEET cSheetName]]]
[AS nCodePage]
```

Status

Slated for after Version 1.0

Remarks

Appends from an ALIAS, or closed DBF file if TYPE is omitted, otherwise appends from the TYPE of external file specified.

Parameters

cFile

Character literal of source file.

FIELDS FieldList

Character literal comma delimited list of fields to append into.

FOR lExpression

Logical expression filtering the records being appened.

DELIMITED

Delimited files are assumed as .txt unless otherwise specified. If DELIMITED WITH is omitted, then the default delimiter is a comma. Character fields are also delimited with double quotes.

DELIMITED WITH Delimiter

Allows specification of a different field character delimiter.

DELIMITED WITH BLANK

Specifies that fields are delimited with blanks instead of commas.

DELIMITED WITH TAB

Specifies that fields are delimited with tabs instead of commas.

DELIMITED WITH CHARACTER Delimiter

Specifies that fields are delimited with the included character literal.

TYPE

Character literal that specifies source file type. The following types are recognized.

Type	Description
CSV	Comma separated values. Like a DELIMITED file, the first row contains field names and is discarded.
SDF	System Data Format file. Fields are a fixed width format in a similar manner to a JAXBase table. Dates are in yyyyMMdd format and DateTime values are in yyyyMMddTHHmmss format.
XLS	An older Excel spreadsheet file.
XLSX	A modern Excel spreadsheet file.
ODS	An Open Document Spreadsheet file.

SHEET cSheetName

Specifies the spreadsheet tab to append from.

AS CodePage (Version 2.0)

Specifies the Code Page of the source file.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
CREATE TABLE TEST_DELIMITED (Name C(30), BirthDate D, Sex C(1), Height N(4,1))
APPEND FROM (_JAXFolder+"samples\import\PEOPLE_DELIMITED.TXT") DELIMITED
USE
```

```
CREATE TABLE TEST_TAB (Name C(30), BirthDate D, Sex C(1), Height N(4,1))
APPEND FROM (_JAXFolder+"samples\import\PEOPLE_TAB.TXT") DELIMITED WITH TAB
USE
```

```
CREATE TABLE TEST_SDF (Name C(30), BirthDate D, Sex C(1), Height N(4,1))
APPEND FROM (_JAXFolder+"samples\import\PEOPLE_SDF.TXT") TYPE SDF
USE
```

```
CREATE TABLE TEST_CSV (Name C(30), BirthDate D, Sex C(1), Height N(4,1))
APPEND FROM (_JAXFolder+"samples\import\PEOPLE.CSV") TYPE CSV
USE
```

```
CREATE TABLE TEST_XLS (Name C(30), BirthDate D, Sex C(1), Height N(4,1))
APPEND FROM (_JAXFolder+"samples\data\PEOPLE.XLS") TYPE XLS
USE
```

```
CREATE TABLE TEST_ODS (Name C(30), BirthDate D, Sex C(1), Height N(4,1))
APPEND FROM (_JAXFolder+"samples\data\PEOPLE_DELIMITED.ODS")
USE
```

Note

You cannot append from a table in a SQL database, but you can SELECT from a SQL table to a cursor and append to a DBF.

You cannot APPEND directly onto a SQL Table. You can SELECT an empty cursor (SELECT * FROM table WHERE 1=2), APPEND to that cursor, and then use the UPDATE FROM Command to copy the cursor into the SQL table.

If a table name is specified and there is no matching alias, then JAXBase will attempt to find the table in the path list if a path was not specified with the name.

Any fields that do not match the FIELDS FieldList parameter are ignored.

It is important to have good error checking in case if a rule is violated in a SQL table, or a CANDIDATE index is violated in a DBF, causing an error to be generated.

See Also

[COPY FILE Command](#)

[COPY TO Command](#)

[EXPORT Command](#)

[GETCP\(\) Function](#)

[IMPORT Command](#)

[SET DEFAULT TO Command](#)

[SET DELETED Command](#)

[SET PATH TO Command](#)

[TRY/CATCH/FINALLY/ENDTRY Command](#)

[UPDATE Method](#)

[SQL Class](#)

APPEND FROM JSON

APPEND FROM JSON cExpression

Remarks

Appends one or more JSON records to a table.

Parameters

cExpression

Character expression containing a valid JASON string.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
cJSON=" [ {Name:"John Smith", Age:20, ShoeSize:10}, {Name:"Jane Doe", Age: 18, ShoeSize:6.5} ]"  
  
CREATE TABLE TEST1 (Name C(20), Age I)  
APPEND FROM JSON cJSON      && Adds two records to TEST1
```

Note

Each JSON record appends a record to the table.

When appending a record, any field names in the JSON string that match field names in the table are used. Extra JSON fields are ignored. Fields with no matching JSON fields are left empty.

The APPEND FROM JSON command will work with JAXBase tables and cursors.

See Also

[GETJSON\(\) Function](#)
[JSONTOOBJ\(\) Function](#)
[OBJTOJSON\(\) Function](#)
[PUTJSON\(\) Function](#)
[UPDATE command](#)

ASSERT

```
ASSERT lExpression [MESSAGE cMessage]
```

Remarks

If the lExpression resolves to true (.T.) then the message is sent to the debug screen and displayed to a dialog with Debug, Cancel, Ignore, and Ignore All buttons.

Selecting one of the buttons produces the following actions:

Debug – Suspends program execution and displays the Trace Window (Version 1.0)

Cancel – Cancels program Execution.

Ignore – Continues program execution.

Ignore All – Performs a SET ASSERT OFF and continues program execution.

Parameters

lExpression

Logical expression which, if evaluated to true (.T.), sends the optional cMessage to the dialog.

cMessage

Optional character expression containing the message to send to the dialog.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
ASSERT .T., "Assert test"
```

Note

If SET ALTERNATE is set to save to a file or printer, then the assert message will be written and what button was selected.

See Also

SET ALTERNATE Command

SET ASSERT Command

SET CONSOLE Command

SUSPEND Command

Trace Window

AVERAGE

```
AVERAGE ExprList [Scope] [FOR lExpr1] [WHILE lExpr2]
    TO VarList | TO ARRAY ArrayName [ADDITIVE]
    [IN nWorkArea | cAlias]
```

Remarks

Takes the averages of the ExprList and puts them into VarList or an array.

Parameters

ExprList

Comma delimited list of expression to use when creating the averages to place into VarList.

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression limiting the records used in AVERAGE while the expression is true.

TO VarList

Character literal list of variable names that will receive the averages.

TO ArrayName [ADDITIVE]

Character literal of array name to create or append to if ADDITIVE is provided.

IN nWorkArea | cAlias

Numeric work area literal or character literal for alias name of table to use.

Example

```
* Demonstrate AVERAGE
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
USE (_JAXFolder+"data\polldata")

* Using a unique numeric literal as one of the expressions provides an
* identifier for each result row. If there are no matching records, all
* non-literal numeric expressions will return 0.00 as their result.
AVERAGE 1,age TO ARRAY aVoteAge for Precinct=203 AND Party="I"
AVERAGE 2,age to ARRAY aVoteAge ADDITIVE for Precinct=203 AND Party="D"
AVERAGE 3,age to ARRAY aVoteAge ADDITIVE for Precinct=203 AND Party="R"

DISPLAY MEMO LIKE aVoteAge
```

Note

When saving to an array, if the array does not exist then it is created. If it does exist, it is overwritten with the corresponding number of elements. If ADDITIVE keyword is included, then the array is appended with a new row. If the number of expressions does not match the existing array column count, an error is generated. If there are no results, the array is updated numeric literals and any non-literal expressions are set to 0.00.

ADDITIVE should not be in the first of a series of AVREAGE statements.

If saving to a variable list, values are returned even if there are no matching records, in which case 0.00 is filled in for any non-literal numeric expression.

See Also

CALCULATE Command

COUNT Command

SUM Command

TOTAL Command

Arrays, Variables, and Objects

BEGIN TRANSACTION

BEGIN TRANSACTION

Status

Slated for Version 0.6

Remarks

Puts the current SQL Database into transaction mode, allowing you to ROLLBACK or COMMIT the changes. JAXBase tables are not affected by transactions.

Note

The COMMIT() and ROLLBACK() functions allow you to manage the transaction directly. If you do not use them, then when the ENDTRANSACTION command is executed, a commit will be sent. If the commit fails, an error will be generated, and the transaction will be automatically rolled back.

The SQL Class can also be used to generate and manage transactions.

SQLTest.prg in the TOOLS folder demonstrates working with a SQL backend database.

See Also

COMMIT() Function

ROLLBACK() Function

ENDTRANSACTION Command

SQL Class

BLANK

```
BLANK [FIELDS FieldList] [Scope] [FOR lExpression1]
      [WHILE lExpression2] [IN nWorkArea | cTableAlias]
```

Remarks

Blanks all fields in a DBF if a field list is not specified.

Parameters

FIELDS

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression

IN nWorkArea | cAlias

Numeric work area literal or character literal for alias name of table to use.

Note

AUTOINC fields are never altered by the BLANK command.

If you execute BLANK with a SCOPE when a CANDIDATE index is open, you risk generating an error.

If a scope is not specified, then only the current record is affected.

See Also

APPEND Command

EMPTY() Function

ISBLANK() Function

REPLACE Command

INDEX Command

BROWSE

```
BROWSE [FIELDS FieldList] [TITLE cTitleText][NOSHOW] [NAME cName] [NOSHOW] [NOWAIT]
```

Status

Slated for Version 1

Remarks

JAXBase has a BrowseWindow class that is used to create a browse-window and can be manipulated like other objects.

PARAMETERS

FIELDS FieldList

An optional list of fields to display in the browse window.

TITLE cTitleText

Optional text to place into the caption of the browse window.

NAME cName

If declared, creates a variable that holds the browse window object and as long as that variable remains in scope, the browse window class can be addressed.

NOSHOW

Prevents the browser window from becoming visible and sets the AUTOONEXIT to false (.F.) so that the browser window stays in memory as long as the variable remains in scope. If a NAME was not specified, the NOSHOW clause is ignored.

NOWAIT

The NOWAIT clause will continue execution after the browser windows is displayed. The NOWAIT clause is meaningless if the NOSHOW clause is also used.

Example

A browse-window can be modified in code by affecting its properties as a JAXBrowseWindow object.

```
BROWSE NAME MyBrowser NOSHOW      && Create the MyBrowser object with Visible=.F.  
  
MyBrowser.Columns[1].Header1="Name"  
MyBrowser.Columns[1].Width=150  
MyBrowser.Columns[2].Header1="Address"  
MyBrowser.Columns[2].Width=150  
MyBrowser.Height=100  
MyBrowser.Width=250  
MyBrowser.ReadOnly=.T.  
MyBrowser.Show
```

Notes

A browse-window's default name is BrowseWindowX, where X is 1 and is incremented until a free name is found.

When the user hits ESCape or otherwise closes a browse window created with a BROWSE command, it automatically unloads from memory. The NAME parameter creates a private variable of that name (unless it already exists in which case it makes that variable a BrowseWindow object) and sets the AUTONEXIT property to a logical false. When AUTOEXIT is set to false, the object remains in memory until it is released, or the variable loses scope.

See Also

APPEND Command
EDIT Command

DRAFT

BUILD

BUILD APPLICATION

```
BUILD APPLICATION cProjectName [TO cApplicationName]
```

Status

Slated for Version 1.0

Remarks

Compiles and bundles all files included in a project into an APP which can be run by the JAXRun executable.

Parameters

cProjectName

Literal filename of the project file (PJX) to compile into an application.

cApplicationName

If supplied, creates an application using the cApplicationName literal.

Example

```
* Create TEST1.APP from Test.pjx  
BUILD APPLICATION Test AS Test1
```

Note

If any errors are generated, a text file named the same as the application name but with an ERR extension is created. This will contain the log of the process and list any errors found.

See Also

[BUILD EXE Command](#)

[BUILD PROJECT Command](#)

[CREATE PROJECT Command](#)

BUILD EXECUTABLE

```
BUILD EXECUTABLE cProjectName [TO cApplicationName] [AS WINDOWS | LINUX]
```

Status

Slated for Version 2.0

Remarks

Compiles and bundles all files included in a project into an executable which can be run natively by the operating system.

Parameters

cProjectName

The name of the project file (PJX) to compile into an application.

cApplicationName

Normally, the application file stem is the same as the application file stem, but the TO clause allows you to specify a different application name.

Windows|Linux

Specify the type of executable for the desired target operating system.

Example

```
* Create a Linux executable  
BUILD EXECUTABLE Test TO Test1 AS LINUX
```

Note

You can create an EXE for any supported operating system.

If any errors are generated, a text file named the same as the application name but with an ERR extension is created. This will contain the log of the process and list any errors found.

See Also

[BUILD APPLICATION Command](#)

[BUILD PROJECT Command](#)

[CREATE PROJECT Command](#)

BUILD PROJECT

BUILD PROJECT cProjectName

Status

Slated for Version 2.0

Remarks

Rebuilds a project, checking for missing references and updating the project's state.

Parameters

cProjectName

The name of the project file (PJX) to rebuild.

Example

BUILD PROJECT Test

Note

If any errors are generated, a text file named the same as the application name but with an ERR extension is created. This will contain the log of the process and list any errors found.

See Also

BUILD APPLICATION Command

BUILD EXECUTABLE Command

CREATE PROJECT Command

DRAFT

CALCULATE

```
CALCULATE eExpressionList [Scope]
  [FOR lExpression1]
  [WHILE lExpression2]
  [TO VarList | TO ARRAY ArrayName] [ADDITIVE]
  [IN nWorkArea | cTableAlias]
```

Remarks

Takes the averages of the ExprList and puts them into VarList or an array.

Parameters

ExprList

Comma delimited list of CALCULATE expressions to use when creating the averages to place into VarList. The list of valid CALCULATE expressions is as follows.

Expression	Result Type	Description
AVE(nExpression)	Numeric	Gets the average of the numeric expression
CNT()	Numeric	Counts the number of records selected
MAX(eExpression)	JAXBase data type	Returns the maximum value of the expression
MIN(eExpression)	JAXBase data type	Returns the minimum value of the expression
NPV(nExpr1, nExpr2[, nExpr3])	Numeric	Computes the net present value of the series
STD(nExpression)	Numeric	Computes the standard deviation
SUM(nExpression)	Numeric	Sums the values of the series
VAR(nExpression)	Numeric	Calculates the variance of the series

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression limiting the records used in AVERAGE while the expression is true.

TO VarList

Character literal list of variable names that will receive the averages.

TO ArrayName [ADDITIVE]

Character literal of array name to create or append to if ADDITIVE is provided.

IN nWorkArea | cAlias

Numeric work area literal or character literal for alias name of table to use.

Example

```

* Demonstrate CALCULATE
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
USE (_JAXFolder+"data\polldata")

* Using the MIN or MAX of a unique numeric literal as one of the
* expressions will provide an identifier for each result row. If
* there are no matching records, all non-literal numeric
* expressions will return 0.00 as their result.
CALCULATE MIN("I"),AVE(age),CNT(Party) TO ARRAY aVoteAge for Precinct=203 AND Party="I"
CALCULATE MIN("D"),AVE(age),CNT(Party) to ARRAY aVoteAge ADDITIVE for Precinct=203 AND Party="D"
CALCULATE MIN("R"),AVE(age),CNT(Party) to ARRAY aVoteAge ADDITIVE for Precinct=203 AND Party="R"

DISPLAY MEMO LIKE aVoteAge

```

Note

When saving to an array, if the array does not exist then it is created. If it does exist, it is overwritten with the corresponding number of elements. If ADDITIVE keyword is included, then the array is appended with a new row. If the number of expressions does not match the existing array column count, an error is generated. If there are no results, the array is updated with non-literal expressions set to 0.00.

ADDITIVE should not be in the first of a series of CALCULATE statements.

If saving to a variable list, values are returned even if there are no matching records, in which case 0.00 is filled in for any non-literal numeric expression.

See Also

- AVERAGE Command
- COUNT Command
- SUM Command
- TOTAL Command
- Arrays, Variables, and Objects

CANCEL

CANCEL

Remarks

Cancels execution of the current program.

Note

If in the IDE, control is returned to the IDE command window. If it is running as an APP or EXE, control is turned back to the operating system.

See Also

SUSPEND
ASSERT
SET STEP

DO CASE ... CASE... OTHERWISE...ENDCASE

```
DO CASE
  CASE lExpression1
    [commands]
  [CASE lExpression2]
    [commands]
  ...
  [OTHERWISE]
ENDCASE
```

Remarks

The CASE structure allows you to specify a block of code to run based on the first logical expression evaluating to True (.T.) or failing that executing the OTHERWISE code block.

Example

```
* User will enter a number and the appropriate
* case or otherwise block will execute
*
SET CONSOLE ON
A=INPUTBOX( "Give me a number between 1 and 5")
I = VAL(A)
CASE
  CASE I=1
    ? "You entered 1"
  CASE I<3
    ? "You entered 2"
  CASE I<6
    ? "You entered ",I
  OTHERWISE
    ? "I said a number between 1 and 5!"
ENDCASE
```

Note

Only the commands following the first CASE clause that evaluates to True (.T.) are executed, or the commands following the DEFAULT if they all fail.

After the selected block of code is executed, control is passed to the command following the ENDCASE statement.

If no case expression evaluates to True (.T.) and there is no OTHERWISE command, then control is passed to the command following the ENDCASE.

See Also

FOR / ENDFOR
 FOREACH / ENDFOR
 IF / ENDIF
 DO / UNTIL
 DO WHILE / ENDDO

CD

CD cLiteral

Remarks

Changes the default folder to an already existing folder.

Parameters

cLiteral

A literal expression such as C:\TEMP

Example

```
* Demonstration of MD command
cFolder="C:\TEMP\JAXBase\Jobs"

* Prevent an error if the folder exists
TRY
MD (cFolder)
CATCH
ENDTRY

=strcmp("TEST","test.txt")

* The following commands all do the same thing
CD C:\TEMP\JAXBase\Jobs
CD &cFolder
CD (cFolder)

* Check to make sure test.txt exists
DIR
```

The default folder is set to C:\TEMP\JAXBase\Jobs and the file listing is displayed in the default console, which will include the file test.txt.

Note

The folder expression must be literal expression, such as `CD C:\TEMP` or a macro expression. If the folder does not exist, an error will be raised.

Windows is not case sensitive for folder and file names, but other operating systems are. You have two choices in this matter. You can use the SET NAMING which will automatically set all file names and paths to either upper, lower, or proper case, or you can be consistent in your naming scheme.

See Also

[ADIR\(\) Function](#)

[MD Command](#)

[SET NAMING Command](#)

[Macros and Macro Expressions](#)

CLEAR

CLEAR

CLEAR

Remarks

Clears the active CONSOLE window.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
? "TESTING"
=INKEY(3)
CLEAR
```

See Also

ACTIVATE CONSOLE Command
SET CONSOLE Command

CLEAR ALL

CLEAR ALL

Remarks

Releases all program variables, arrays, and objects, and classes. It also closes all editor windows, open tables, files, and any active projects.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

MODIFY FILE TEST.TXT NOWAIT

A=4
B=7
DISPLAY MEMORY

CLEAR ALL
DISPLAY MEMORY
```

CLEAR CLASS

CLEAR CLASS [cClassName | ALL]

Status

Slated for Version 1

Remarks

Clears all references to the named class from memory.

Parameters

cClassName

Literal expression of the class to clear from memory.

ALL

Causes all class references and class libraries to be removed from memory.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=CREATEOBJECT( "form" )

DISPLAY MEMO

CLEAR CLASS FORM
DISP MEMO
```

Note

Classes and class libraries opened with MODIFY CLASS are not affected by the CLEAR CLASS command.

If a class name is not provided, the command is ignored.

CLEAR CONSOLE

```
CLEAR CONSOLE [cConsoleName]
```

Remarks

Like the CLEAR command but specifies which console to clear. If the named console does not exist, an error is generated.

Parameters

cConsoleName

Literal name expression indicating console to clear.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR CONSOLE
```

Note

If a console name is not provided, the DEFAULT console is cleared.

In Windows, only the DEFAULT console can be created. Version 2.0 is expected to address this shortcoming.

CLEAR CLASSLIB - !!!

```
CLEAR CLASSLIB [cClassLibraryName]
```

Status

Slated for Version 1

Remarks

Clears all references of the specified class library from memory and closes the class library file if it is open.

Parameters

cClassLibararyName

Literal file name of class library to open.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

Note

A VCX extension is assumed for the cClassLibrary literal.

If a name is not provided, an open dialog will prompt you for a class library to open.

CLIB_EDITOR.APP in the TOOLS folder is used to modify class libraries.

All user defined classes are either manually created in code or stored in class libraries.

CLEAR EVENTS

CLEAR EVENTS

Status

Slated for Version 1

Remarks

Clears the active READ EVENTS from memory, releasing control to the code after the READ EVENTS command.

Example

```
* Demonstrate CLEAR EVENTS  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
aFrm=createobject("form")  
aFrm.AddObject("TMR1","timer")  
aFrm.TMR1.Enabled=.T.      && Timer initializes when form is created  
aFrm.TMR1.Interval=5000  
aFrm.SetMethod("TIMER","CLEAR EVENTS")  
aFrm.Show  
READ EVENTS
```

CLEAR ERRORS

CLEAR ERRORS

Remarks

Clears all errors from the error stack.

DRAFT

CLEAR MACROS

CLEAR MACROS

Status

Slated for Version 1

Remarks

Clears all ON KEY LABEL macros from memory.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
ON KEY LABEL F5 ? "HELLO"  
DISPLAY MACROS
```

```
CLEAR MACROS  
DISPLAY MACROS
```

CLEAR MEMORY

CLEAR MEMORY

Status

Slated for Version 1

Remarks

Releases all program variables, arrays, macros, and objects from memory.

DRAFT

CLEAR PROGRAM

CLEAR PROGRAM [cProgramName]

Status

Slated for Version 1

Remarks

Clears the named program from the cache.

Parameters

cProgramName

Literal file name or file stem of program to clear from memory.

Note

If no program name is specified, it clears the program cache, including any cached classes.

CLEAR TYPEAHEAD

CLEAR TYPEAHEAD

Status

Slated for Version 1

Remarks

Clears the keyboard buffer.

DRAFT

CLOSE

CLOSE ALL

CLOSE ALL

Status

Slated for Version 1

Remarks

Close the alternate file and all open editors, projects, databases, tables, and indexes.

DRAFT

CLOSE ALTERNATE

CLOSE ALTERNATE

Status

Slated for Version 1

Remarks

Closes the ALTERNATE output device. If none specified, it is ignored.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET ALTERNATE TO TEST.TXT
SET ALTERNATE ON

DIRECTORY (JAXFOLDER+"TOOLS\*.*")

CLOSE ALTERNATE
```

Note

CLOSE ALTERNATE is the same as executing the commands

```
SET ALTERNATE OFF
SET ALTERNATE TO
```

CLOSE DATABASE

CLOSE DATABASE cDatabaseName | ALL

Status

Slated for Version 1

Remarks

Close the current active database.

Parameters

cDatabaseName

Name of database to close.

ALL

If ALL is used, then all active databases are closed.

Note

When a database is closed, all active transactions are abandoned and data is discarded.

If the cDatabaseName is not open, an error message is generated.

CLOSE DATASESSION

CLOSE DATASESSION nID | ALL

Status

Slated for Version 1

Remarks

Close the current data session, the indicated data session ID, or all user data sessions.

Parameters

nExpression

Numeric literal indicating which data session to close.

ALL

Close all active user data sessions.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
* Initialize and close data session 2
SELECT DATASESSION 2
A=2
CLOSE DATASESSION (A)
```

CLOSE INDEXES

CLOSE INDEXES

Status

Slated for Version 1

Remarks

Close all indexes that are open in the current workarea. If none are open, it is ignored.

DRAFT

CLOSE TABLES

CLOSE TABLES [ALL]

Status

Slated for Version 1

Remarks

Close all tables and cursors in the current data session.

Parameters

ALL

If ALL is specified, closes all tables and cursors in all data sessions. Open databases are not affected.

COMPILE

COMPILE cFileName

Remarks

Compiles source files and definitions into JAXBase runtime files.

Parameters

cFileName

Literal filename of the file to compile. If no extension is provided, then PRG is assumed.

Example

COMPILE TEST.*

Notes

The file extension must match what's being compiled if you are compiling a FORM, MENU, POPUP, BAR, or CLASS LIBRARY created with their respective builders, otherwise an error will be raised.

The compile command uses the extension of the specified file to determine what type of file is being compiled. If you have multiple file stems with different file extensions and do not specify the type of file to compile, then all will be compiled.

If you use a wildcard in cFileName then all matching files will be compiled.

See Also

[CREATE COMMAND Command](#)

[CREATE CLASS Command](#)

[CREATE FORM Command](#)

CONTINUE

CONTINUE

Remarks

Example

```
SET TALK OFF
CLOSE DATABASES
USE customers  && Open Customer table

STORE 0 TO gnCount

LOCATE FOR ALLTRIM(UPPER(customers.country)) = 'CANADA'

* All records are displayed where the country is equal to CANADA.
DO WHILE FOUND()
    gnCount = gnCount + 1
    ? company,address,city,country
    CONTINUE
ENDDO

? 'Total companies: '+ LTRIM(STR(gnCount))
```

Notes

After a successful LOCATE, the CONTINUE command looks for the next match based on the same criteria as the LOCATE and set the FOUND(), EOF(), and RECNO() accordingly.

A CONTINUE that is issued after a failed LOCATE/CONTINUE set the record pointer past the end of the file and set FOUND() to .F. (false), EOF() to .T. (true), and RECNO() to record count + 1.

CONTINUE can be repeated endlessly without generating an error, using FOUND() will allow you to detect if it was successful.

See Also

[EOF\(\) Function](#)
[FOUND\(\) Function](#)
[LOCATE Command](#)
[SEEK Command](#)

COPY FILE

```
COPY FILE cFileName1 TO cFileName2
```

Status

Slated for Version 1

Remarks

Copies a file.

Parameters

cFileName1

Literal filename which is the source.

cFileName2

Literal filename which is the destination.

Note

COPY FILE will make an exact duplicate of cFileName1 to cFileName2.

If cFileName already exists, it will be overwritten.

If a problem occurs during copy, an appropriate error is generated.

See Also

RENAME Command

COPY INDEXES

```
COPY INDEXES cIDXList | All [TO cCDXfile]
```

Status

Possible integration into for Version 2.0

Remarks

Used to copy one or more single index file into a compound index file.

DRAFT

COPY STRUCTURE

```
COPY STRUCTURE TO cTableTo [FIELDS FieldList] [DATABASE cDBName] [OVERWRITE]
```

Status

Slated for Version 1

Remarks

Copy the currently open table structure to a new DBF or SQL Table.

Parameters

cTableTo

Character literal of target table.

FIELDS FieldList

Optional comma delimited character field literals.

DATABASE cDBName

Optional character literal of database alias to send the new structure to create a SQL table.

OVERWRITE

Optional flag indicates if cTableTo exists, to overwrite it.

Example

```
* Demonstrate COPY STRUCTURE by copying  
* the supplied polldata table to a file  
* in the default directory.
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
USE (_JAXFolder+"data\polldata")  
COPY STRUCTURE TO pol12 OVERWRITE  
USE pol12  
LIST STRUCTURE
```

Note

If the destination table exists and OVERWRITE is not included, an error is generated.

JAXBase converts the fields in a specific manner, depending on the backend database engine and may not create the best structure for SQL use.

See Also

[COPY TABLE Command](#)

[CREATE TABLE Command](#)

[SQL Class](#)

COPY TO ARRAY

```
COPY TO ARRAY ArrayName
  [FIELDS cFieldList | FIELDS LIKE cSkeleton | FIELDS EXCEPT cSkeleton]
  [Scope] [FOR lExpression1] [WHILE lExpression2]
```

Status

Slated for Version 1

Remarks

Copy fields from records that fit the scope and logical expressions to an array.

Parameters

ArrayName

Name of array to create.

FIELDS FieldList

Specifies the fields to populate from the array.

FIELDS LIKE

Specifies a field skeleton where matching fields are populated from the array.

FIELDS EXCEPT

Specifies fields to skip when populating from an array

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT *nRecords*, RECORD *nRecordNo*, REST, and TOP *nRecords*. The default scope is ALL.

FOR *lExpr1*

Logical expression used to filter what records are used.

WHILE *lExpr2*

Logical expression limiting the records used in AVERAGE while the expression is true.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
USE (_JAXFolder+"data\polldata")
COPY TOP 10 TO ARRAY aTop10
DISPLAY MEMORY LIKE aTOP10
```

Note

If ArrayName does not exist, it will be created with the corresponding number of columns and rows. If the ArrayName exists, it will be overwritten. If no records fit the parameters, then the ArrayName is not created or altered.

The number of rows you can create in an array is subject to memory.

See Also

APPEND FROM ARRAY Command

DIMENSION Command

GATHER Command

STORE Command

DRAFT

COPY TO

Copy to a DBF

```
COPY TO cFile [FIELDS FieldList]
[Scope] [FOR lExpression1] [WHERE lExpression2]
[AS nCodePage]
```

Copy to a Delimited File

```
COPY TO cFile [FIELDS FieldList]
[Scope] [FOR lExpression1] [WHERE lExpression2]
[DELIMITED [WITH Delimiter | WITH BLANK | WITH TAB | WITH CHARACTER Delimiter]]
[AS nCodePage]
```

Copy to a File Type

```
COPY TO cFile [FIELDS FieldList]
[Scope] [FOR lExpression1] [WHERE lExpression2]
[TYPE CSV| SDF | [XLS | XLSX | ODS]
[AS nCodePage]
```

Status

Slated for after Version 1.0

Remarks

Creates a new file from the contents of the currently selected file.

Parameters

cFile

Character literal of source file.

FIELDS FieldList

Character literal comma delimited list of fields to append into.

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression limiting the records used in AVERAGE while the expression is true.

DELIMITED

Delimited files are assumed as .txt unless otherwise specified. If DELIMITED WITH is omitted, then the default delimiter is a comma. Character fields are also delimited with double quotes.

DRAFT

DELIMITED WITH Delimiter

Allows specification of a different field character delimiter.

DELIMITED WITH BLANK

Specifies that fields are delimited with blanks instead of commas.

DELIMITED WITH TAB

Specifies that fields are delimited with tabs instead of commas.

DELIMITED WITH CHARACTER Delimiter

Specifies that fields are delimited with the included character literal.

TYPE

Character literal that specifies source file type. The following types are recognized.

Type	Description
CSV	Comma separated values. Like a DELIMITED file, the first row contains field names and is discarded.
SDF	System Data Format file. Fields are a fixed width format in a similar manner to a JAXBase table. Dates are in yyyyMMdd format and DateTime values are in yyyyMMddTHHmmss format.
XLS	An older Excel spreadsheet file.
XLSX	A modern Excel spreadsheet file.
ODS	An Open Document Spreadsheet file.

AS CodePage (Version 2.0)

Specifies the Code Page of the source file.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
COPY TO TEST_XLS TYPE XLS FOR Precinct=203
```

Note

If an error occurs trying to write the file, an error is generated.

If the destination file exists, it will be overwritten.

Writing to a worksheet writes the data to the default sheet name.

A CSV file will have the JAXBase field names in the first row with character data delimited with quotes.

MEMO, GENERAL, or BLOB fields will not be written, but any valid expression that evaluates to less than 255 characters will be written. Expressions 255 characters or greater will generate an error.

See Also

APPEND FROM Command

COY FILE Command

EXPORT Command

IMPORT Command

DRAFT

COUNT

COUNT [SCOPE] [FOR lExpression1] [WHILE lExpression2] TO Var

Remarks

The COUNT command counts the number of matching records within the scope and returns the value to Var. If no records match, the variable is set to 0.00.

Parameters

Scope

Specifies a range of records to be scanned and only the records within the range are included. The valid scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, REST, and TOP nRecords. The Scope allows you to limit the number of records processed.

FOR

A logical expression that must be evaluated to True (.T.) for the current record to be processed by the block of code. If the FOR command is not present, it is assumed to be True (.T.).

WHILE

A logical expression that must be evaluated to True (.T.) for processing to continue. If the expression evaluates to False (.F.) then control passes to the command after the ENDSCAN.

TO Var

Character literal representing the variable name to which the count is saved.

Example

```
* Demonstrate CALCULATE
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
USE (_JAXFolder+"data\polldata")

a=0
SCAN FOR PRECINCT=203
    a=a+1
ENDSCAN

? a, "records counted"
```

Note

Both the FOR and WHILE commands are evaluated each time the SCAN command is processed.

ALL is the default SCOPE value one is not provided.

When the ENDSCAN command is executed, the table that was current when entering the SCAN/ENDSCAN loop is reselected, and the record pointer advanced, before passing control back to the SCAN command.

See Also

TOTAL Command

Scope Clauses

DRAFT

CREATE TABLE / CURSOR

```
CREATE TABLE cFile from ARRAY ArrayName

CREATE TABLE cFile
  (cField cType [(nWidth [,nPrecision])]
  [, cField cType [(nWidth [,nPrecision])]
  [, cField...])
```

Slated for Version 2.0

```
CREATE TABLE cFile
  (cField cType [(nWidth [,nPrecision])] [NULL | NOTNULL] [ENCRYPTED] [NOCPTRANS]
  [AUTOINC [NEXTVALUE nNext [STEP nStep]]]
  [, cField cType [(nWidth [,nPrecision])] [NULL | NOTNULL] [ENCRYPTED] [NOCPTRANS]
  [, cField...]) [CODEPAGE nCodePage] [COLLATE cCollateSequence] [ENCRYPTED]
```

Remarks

Create a JAXBase DBF table.

Parameters

cFile

Character literal name of the table to create.

cField

Character literal name of a field in the table.

cType

Character literal type code (see below).

See AFIELDS() Function.

nWidth

Numeric literal for width of the field.

nPrecision

Numeric literal for how many places to the right of the decimal point are supported (for numeric non-Integer fields).

NULL | NOT NULL (Version 2.0)

Indicates if the field can hold a .NULL. value. (Version 2.0)

ENCRYPTED (Version 2.0)

Indicates that encryption should be used when writing to the table. (Version 2.0)

NOCPTRANS (Version 2.0)

Indicates that the field should not allow Code Page translation. (Version 2.0)

AUTOINC [NEXTVALUE nNext [STEP nStep]] (Version 2.0)

The field is autoincrementing and by default has a nNext value of 1 and nStep value of 1 if nNext or nStep are omitted. AUTOINC values are integer only and can range from -2,147,483,647 to 2,147,483,647.

CODEPAGE nCodePage (Version 2.0)

Specifies the code page to use for this table. See Code Page Support

COLLATE cCollateSequence (Version 2.0)

Specifies the collation sequence other than the default of MACHINE.

See Supporting International Applications

FROM ARRAY ArrayName

Array name holding field information in the following format.

See AFIELDS() Function

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
CREATE TABLE TEST1 (Name C(20), Age I)  
DISPLAY STRUCTURE
```

Note

If a path is not specified in cFile then the table is placed in the default path.

The new table will be opened in the lowest available work area.

nStep cannot be a value under 1.

A table can hold up to 254 fields.

See Also

AFIELDS() Function

DISPLAY STRUCTURE Command

USE Command

Code Page Support

Supporting International Applications

CREATE DATABASE

```
CREATE DATABASE cDBName [USING cConnection | oDbObject]
```

Status

Slated for Version 1

Remarks

Create a new database to a SQL backend.

Parameters

cDBName

Character literal of the database name to create.

cConnection

String expression with a valid SQL Connection string to the desired server.

oDbObject

Object created using the SQL Class.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

MyDB=CREATEOBJECT( "sql" )

* Correct the connection data
WITH MyDB
    .Driver="SQL Server"
    .Server="MyServer"
    .Port=1433    && Default SQL Server connection
    .USERID="sqladmin"
    .PASSWORD="BobIsAtHD-3167"
    .DATABASE="MASTER"
ENDWITH

CREATE DATABASE USING MyDB
```

Note

The oDbObject does not have to be connected, but the server information must be valid.

If any problem arises while creating the database, the appropriate error is generated.

See Also

OPEN DATABASE Command

SQL Class

CREATE FORM

```
CREATE FORM [cFormName] [AS cClassName FROM cClssLibrary | cDEFFile]
```

Status

Slated for Version 1.0

Remarks

Opens the Form Editor and creates a new SCX file named after cFormName.

Parameters

cFormName

Character literal of the form name to create.

cClassName

Character literal of the class to copy from cClassLibrary or DEF file.

cClassLibrary

Character literal name of the VCX class library to get the cClassName definition.

cDEFFile

Name of DEF file that holds cClassName

Note

If the base class of cClassName is not a form, an error is generated.

See Also

COMPILE Command

DO FORM Command

MODIFY FORM Command

Form Designer

CREATE LABEL

```
CREATE LABEL cLabel [AS cClassName FROM cClassLibrary | cDEFFile]
```

Status

Slated for after Version 1.0

Remarks

Open the label editor and create a new LBX file.

Parameters

cLabel

Character literal name of the label LBX file to create.

cClassName

Character literal of the class to copy from cClassLibrary or DEF file.

cClassLibrary

Character literal name of the VCX class library to get the cClassName definition.

cDEFFile

Name of DEF file that holds cClassName

Note

If the cLabel file already exists, it is overwritten.

See Also

MODIFY LABEL Command

LABEL Class

Label Designer

CREATE MENU

```
CREATE MENU cMenu [AS cClassName FROM cClassLibrary | cDEFFile]
```

Status

Slated for after Version 1.0

Remarks

Open the menu editor and create a new MNX file.

Parameters

cMenu

Character literal name of the label MNX file to create.

cClassName

Character literal of the class to copy from cClassLibrary or DEF file.

cClassLibrary

Character literal name of the VCX class library to get the cClassName definition.

cDEFFile

Name of DEF file that holds cClassName

Note

If the cMenu file already exists, it is overwritten.

See Also

MODIFY MENU Command

MENU Class

Menu Designer

CREATE PROJECT

CREATE PROJECT cProject

Status

Slated for Version 1.0

Remarks

Open the project editor and create a new PJX file.

Parameters

cProject

Character literal name of the label PJX file to create.

Note

If the cProject file already exists, it is overwritten.

See Also

MODIFY PROJECT Command

PROJECT Class

Project Designer

CREATE QUERY

CREATE QUERY cQuery

Status

Slated for after Version 1.0

Remarks

Open the Query editor and create a new QRX file.

Parameters

cQuery

Character literal name of the label QRX file to create.

Note

If the cQuery file already exists, it is overwritten.

See Also

MODIFY QUERY Command

QUERY Class

Query Designer

CREATE REPORT

```
CREATE REPORT cReport [AS cClassName FROM cClassLibrary | cDEFFile]
```

Status

Slated for after Version 1.0

Remarks

Open the report editor and create a new RPX file.

Parameters

cReport

Character literal name of the label RPX file to create.

cClassName

Character literal of the class to copy from cClassLibrary or DEF file.

cClassLibrary

Character literal name of the VCX class library to get the cClassName definition.

cDEFFile

Name of DEF file that holds cClassName

Note

If the cReport file already exists, it is overwritten.

See Also

[MODIFY Report Command](#)

[REPORT Class](#)

[Reprt Designer](#)

DEBUG

DEBUG

Status

Slated for after Version 1.0

Remarks

Opens the JAXBase Debugger Window

See Also

DEBUGOUT Command

SET STEP Command

DEBUGOUT

DEBUGOUT [TO cConsole] eExpression1 [,eExpression2 ...]

Status

Slated for Version 1

Remarks

Send information out the active or designated console.

Parameters

cConsole

Name of console to send the expressions to.

eExpression1 [,eExpression2 ...]

Comma delimited list of expressions to send to the console.

Note

Versions below Version 2.0 will only have the DEFAULT console.

If the cConsole name is not open, an error will be generated.

If TO cConsole is not specified, the active console is used.

See Also

SET CONSOLE Command

ACTIVATE CONSOLE Command

DEFINE CLASS

```
DEFINE CLASS ClassName1 AS ParentClass
  [[PROTECTED | HIDDEN] PropertyName1, PropertyName2 ...]
  [[.]Object.]PropertyName = eExpression ...
  [ADD OBJECT [PROTECTED] ObjectName AS ClassName2 [NOINIT] [WITH cPropertylist]]
  [[PROTECTED | HIDDEN] FUNCTION | PROCEDURE Name[_ACCESS | _ASSIGN]
    ([cParamName | cArrayName[])
      cStatements
    [ENDFUNC | ENDPROC]
  ENDDEFINE
```

Status

Slated for Version 1.0

Remarks

Define a class using JAXBase source code instead of the class editor.

Example

```
* Program: LoadForm.prg
thisForm = CREATEOBJECT( "MyForm" )
thisForm.show()

DEFINE CLASS MyForm AS FORM
  Height=600
  Width=800
  Caption="My First Form"
  WindowType=1 // Modal

  ADD OBJECT btnOK AS Button with caption="\<OK",left=700,top=640
  btnOK.ADDCODE("click")="wait window this.name timeout 5"
ENDDEFINE
```

Note:

The order of precedence for loading a user defined class definition is:

DEFINE CLASS definitions in current PRG

Class libraries opened using SET CLASSLIB command

DEFINE CLASS definitions in modules opened with SET PROCEDURE

DEF file where the stem of file name is same as the class name

You can use the JAXBase ADDCODE("method/event name") method to assign a method or event with JAXBase source code. The code is compiled and then placed into the method or event during runtime and will be executed the next time that method or event is called. If code already exists in that method or event, an error will be raised.

When loaded, a class definition is put into a cache and executed when the class is referenced, creating an object as defined. The CREATEOBJECT command is used to convert the class definition into an object.

ClassDemo.prg in the SAMPLES folder shows how to define and use classes.

DELETE

```
DELETE [Scope] [FOR lExpression1] [WHILE lExpression2] [IN nWorkArea | cAlias]
```

Remarks

Marks the specified records in the indicated work area or alias as deleted.

Parameters

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpression1

Logical expression used to filter what records are used.

WHILE lExpression2

Logical expression limiting the records searched to while the expression is true.

IN nWorkArea | cAlias

Numeric work area literal or character literal for alias name of table to use.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
CREATE TABLE Test1 (Name C(20))
INSERT INTO TEST1 (NAME) VALUES ("JOHN SMITH")
INSERT INTO TEST1 (NAME) VALUES ("JANE SMITH")
INSERT INTO TEST1 (NAME) VALUES ("JOHN DOE")
INSERT INTO TEST1 (NAME) VALUES ("JANE DOE")
```

```
DELETE ALL
COUNT TO A FOR DELETED()
? A, "deleted records"
```

See Also

COUNT Command
CREATE TABLE Command
DELETE Command
DELETED() Function
INSERT Command

DELETE FILE

DELETE FILE cFileName

Status

Slated for Version 1.0

Remarks

Deletes the indicated file from storage.

Parameters

cFileName

Character literal file name to delete.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

STRTOFILE( "HELLO!", "test.txt", 0 )  && Overwrite if it exists
DELETE FILE test.txt

IF FILE( "test.txt" )
    ? "DELETE FAILED"
ENDIF
```

Note

If the file does not exist, the command is ignored, otherwise if there is a problem with deleting the file, an error is generated.

You may use wildcards in the name to delete more than one file at a time.

See Also

ADIR() Function

DIR / DIRECTORY Command

CREATE FILE Command

MODIFY FILE Command

DIMENSION

```
DIMENSION ArrayName1(nRows1 [,nColumns1]) [AS cType]
    [,ArrayName2(nRows1 [,nColumns1])[AS cType]...]
```

Remarks

Create one or more arrays with optional type locking.

Parameters

ArrayName1

nRows1 [,nColumns1]

Numeric expressions that define how many rows and columns with which to initialize the array.

AS cType

You may specify that all elements of an array to be of a type. The allowed data types are as follows.

Code	Full Name	Compatible Value Types
B	Double	Double, Float, Integer, Numeric, Currency
C	Character	Character, VarChar, VarBinary, Memo, General, or Blob
D	Date	Date or DateTime
F	Float	Double, Float, Integer, Numeric, Currency
I	Integer	Double, Float, Integer, Numeric, Currency
L	Logical	Logical, numeric 1 (.T.) and 0 (.F.)
N	Numeric	Double, Float, Integer, Numeric, Currency
O	Object	Any JAXBase Class Object
T	DateTime	Date or DateTime
V	VarChar	Character, VarChar, VarBinary, Memo, General, Blob
Y	Currency	Double, Float, Integer, Numeric, Currency

Example

```
* Demonstrate DIMENSION
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

DIMENSION aArray1[2,2] AS Double, aArray2[2,2] AS DATE, aArray3[2,2]
DISPLAY MEMORY LIKE aArray?
```

Note

All array elements are initialized to false(.F.) unless AS cType is used, in which case the empty value for that type is created.

Arrays set AS cTYPE that receive compatible data types will convert as needed, otherwise an error will be generated.

An array that is not set AS cType can hold any data type in any array element.

Arrays passed as parameters via DO / WITH are passed by reference unless they are part of an expression or you are sending one element of the array, in which case that parameter is sent by value.

See Also

APARAMETERS Command
LOCAL Command
LPARAMETERS Command
PRIVATE Command
PARAMETERS Command
PUBLIC Command

DRAFT

DIRECTORY / DIR

```
Directory [cSkeleton] [TO PRINTER [PROMPT] | TO FILE cFileName [ADDITIVE]]
```

Remarks

List the directory contents to the active console, printer, or to a text file.

Parameters

cSkeleton

Character literal of filename with or without wild card characters.

TO PRINTER [PROMPT]

Sends the directory information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [ADDITIVE]

Sends the directory information out to the cFileName defined with a character literal. If ADDITIVE is included, the directory is append to the file, otherwise it is overwritten.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
DIR (_JAXHome+"*\.*")
```

Note

You can add path information to the cSkeleton and cFileName.

The ? and * wildcards can be used in cSkeleton, but not cFileName.

If no files match cSkeleton, then a message indicating no files will be sent to the output device.

If an error occurs while trying to list the directory information, such as media, network, or printer error then an error will be generated.

See Also

ADIR() Function

DISPLAY DATABASE

```
DISPLAY DATABASE [ TO PRINTER [PROMPT] | TO FILE fileName [ADDITIVE]] [NOCONSOLE]
```

Status

Slated for Version 1.0

Remarks

Displays a database to console, printer, or filename.

Parameters

TO PRINTER [PROMPT]

Sends the directory information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [ADDITIVE]

Sends the directory information out to the cFileName defined with a character literal. If ADDITIVE is included, the directory is appended to the file, otherwise it is overwritten.

NOCONSOLE

Prevents information going to a printer or file from also displaying to the active console.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
* Assumes a SQL Connection exists to a backend database
OPEN DATABASE TEST USING FILETOSTR("SQLConnect.dsn")
```

```
DISPLAY DATABASE TO FILE dbinfo.txt NOCONSOLE
```

Note

If displaying to console, then the display pauses after a screen full and you must press a key or click a mouse button to continue. Pressing the ESC or CTRL+C key cancels the operation.

If displaying to a printer or file and NOCONSOLE is not included, the information will also go to the active console (if one exists), but the display will not pause.

See Also

DISPLAY Commands

LIST Commands

DISPLAY FILES

```
DISPLAY FILES [cSkeleton] [TO PRINTER [PROMPT] | TO FILE cFileName [ADDITIVE]]
```

Remarks

Display the directory contents to the active console, printer, or to a text file.

Parameters

cSkeleton

Character literal of filename with or without wild card characters.

TO PRINTER [PROMPT]

Sends the directory information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [ADDITIVE]

Sends the directory information out to the cFileName defined with a character literal. If ADDITIVE is included, the directory is append to the file, otherwise it is overwritten.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
DISPLAY FILES (_JAXHome+"*.*")
```

Note

If the output device is a console, then the directory is paused after every screen-full and you must press a key or click a mouse button to continue. If the ESC key is pressed, output stops.

You can add path information to the cSkeleton and cFileName.

The ? and * wildcards can be used in cSkeleton, but not cFileName.

If no files match cSkeleton, then a message indicating no files will be sent to the output device.

If an error occurs while trying to list the directory information, such as media, network, or printer error then an error will be generated.

See Also

ADIR() Function

DISPLAY MEMORY

```
DISPLAY MEMORY [LIKE cSkeleton]
    [TO PRINTER [PROMPT] | TO FILE fileName [ADDITIVE]] [NOCONSOLE] [EXTENDED]
```

Status

Slated for Version 1

Remarks

Displays all user JAXBase or user defined variables, arrays, and objects.

Parameters

LIKE cSkeleton

Specifies a variable skeleton containing wild cards is used to match the variables to display.

TO PRINTER [PROMPT]

Sends the variable information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [ADDITIVE]

Sends the variable information out to the cFileName defined with a character literal. If ADDITIVE is included, the information is append to the file, otherwise it is overwritten.

[EXTENDED]

Normally, an object will just be listed as an object. Extended will also list the object properties at the time.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
DISPLAY MEMO LIKE _JAX* TO FILE JAXInfo.txt EXTENDED
```

Note

The JAXBase environment variables will not display unless you use a cSkeleton such * or as _*.

If the output device is a console, then the information display is paused after every screen-full and you must press a key or click a mouse button to continue. If the ESC key is pressed, output stops.

See Also

CREATEOBJECT() Function
DIMENSION Command
DISPLAY Commands
LIST Commands

DRAFT

DISPLAY OBJECTS

```
DISPLAY OBJECTS [LIKE cObjectSkeleton]
    [TO PRINTER [PROMPT] | TO FILE fileName [ADDITIVE]] [NOCONSOLE]
```

Status

Slated for Version 1.0

Remarks

Displays all active user defined objects in memory to a device.

Parameters

LIKE cSkeleton

Specifies a variable skeleton containing wild cards is used to match the variables to display.

TO PRINTER [PROMPT]

Sends the variable information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [ADDITIVE]

Sends the variable information out to the cFileName defined with a character literal. If ADDITIVE is included, the information is append to the file, otherwise it is overwritten.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
DISPLAY OBJECTS LIKE *
```

Note

Normally DISPLAY OBJECTS will not display JAXBase environmental objects, but If you use a cSkeleton with a wildcard that would include the JAXBase environment objects, then they will also be displayed.

Each object will also have its properties at the time listed to the desired device.

If the output device is a console, then the directory is paused after every screen-full and you must press a key or click a mouse button to continue. If the ESC key is pressed, output stops.

See Also

CREATEOBJECT()
DISPLAY Commands
LIST Commands

DRAFT

DISPLAY STATUS

DISPLAY STATUS [TO PRINTER [PROMPT] | TO FILE fileName [ADDITIVE]] [NOCONSOLE]

Status

Slated for Version 1.0

Remarks

Displays the JAXBase status to console, printer, or file.

Parameters

TO PRINTER [PROMPT]

Sends the directory information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [ADDITIVE]

Sends the directory information out to the cFileName defined with a character literal. If ADDITIVE is included, the directory is appended to the file, otherwise it is overwritten.

NOCONSOLE

Prevents information going to a printer or file from also displaying to the active console.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
DISPLAY STATUS
```

Note

If displaying to console, then the display pauses after a screen full and you must press a key or click a mouse button to continue. Pressing the ESC or CTRL+C key cancels the operation.

If displaying to a printer or file and NOCONSOLE is not included, the information will also go to the active console (if one exists), but the display will not pause.

DISPLAY STATUS can be very helpful when debugging a problem in your code.

The DISPLAY STATUS command lists out the following information about the JAXBase Environment.

Data Environment

- Open data sessions
 - o Open tables or cursor names/aliases
 - Open indexes for each table
- Open table information
 - o Open index information for each table/Cursor
- Active SQL Objects
- Active Database Connections

Program Environment

- Current File/Procedure being executed
- Current Line and Source line (if available)
- _JAX class properties
- _CPU class properties
- _Drives array
- _Monitors array
- _PRINTERS information
- Current data session
- Current work area
- Current SET Command settings
- Active keyboard macros
- User defined variables, arrays, and objects

See Also

- DISPLAY Commands
- LIST Commands

DISPLAY STRUCTURE

```
DISPLAY STRUCTURE [IN nWorkArea | cAlias]
    [TO PRINTER [PROMPT] | TO FILE cFileName [ADDITIVE]] [NOCONSOLE]
```

Status

Slated for Version 1.0

Remarks

Displays the structure of an open JAXBase DBF or cursor.

Parameters

IN nWorkArea | cAllias

Numeric work area literal or character literal for alias name of table to use.

TO PRINTER [PROMPT]

Sends the directory information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [**ADDITIVE**]

Sends the directory information out to the cFileName defined with a character literal. If ADDITIVE is included, the directory is append to the file, otherwise it is overwritten.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
DISPLAY STRUCTURE
```

Note

If displaying to console, then the display pauses after a screen full and you must press a key or click a mouse button to continue. Pressing the ESC or CTRL+C key cancels the operation.

If displaying to a printer or file and NOCONSOLE is not included, the information will also go to the active console (if one exists), but the display will not pause.

See Also

[AFIELDS\(\) Function](#)

[DISPLAY Commands](#)

[LIST Commands](#)

DISPLAY TABLES

```
DISPLAY TABLES [DATABASE [cDbSkeleton]] [DATASESSION nID]
    [TO PRINTER [PROMPT] | TO FILE cFileName [ADDITIVE]] [NOCONSOLE]
```

Status

Slated for Version 1.0

Remarks

Displays all open tables in a data session or tables in a database.

Parameters

DATABASE [cDbSkeleton]

If included, displays the tables any open database matching cDbSkeleton. If cDbName is not included, lists the tables in the current database.

DATASESSION nID

If included, displays the tables for the data session, otherwise just the tables in the current data session.

TO PRINTER [PROMPT]

Sends the directory information out to the active printer or to a printer you choose if PROMPT is included.

TO FILE cFileName [ADDITIVE]

Sends the directory information out to the cFileName defined with a character literal. If ADDITIVE is included, the directory is append to the file, otherwise it is overwritten.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
SELECT 1
USE (_JAXFolder+"data\phonebook")

DISPLAY TABLES
```

Note

If displaying to console, then the display pauses after a screen full and you must press a key or click a mouse button to continue. Pressing the ESC or CTRL+C key cancels the operation.

If displaying to a printer or file and NOCONSOLE is not included, the information will also go to the active console (if one exists), but the display will not pause.

If a database is not open and DATABASE is included without a cDbSkeleton, an error message will be generated. If cDbSkeleton does not contain a wildcard (*) or (?) and there is no match, an error message will be generated.

See Also

ASESSIONS() Function
DISPLAY Commands
LIST Commands
OPEN DATABASE Command
USE Command

DRAFT

DO

DO Program

```
DO cProgram1 | cProcedure [IN cProgram2] [WITH ParameterList]
```

Remarks

Executes a JAXBase program or procedure.

Parameters

cProgram1

Specifies the name of the program to execute.

cProcedure

Specifies the name of the procedure to execute.

IN cProgram2

Executes a procedure in the program file specified with ProgramName2. When the file is located, the procedure is executed. If the program file cannot be located or the procedure cannot be located in the program file, an error is generated.

WITH ParameterList

Specifies parameters to pass to the program or procedure.

Note

If you do not include an extension with cProgram1, JAXBase looks for and executes versions of the program in the following order: EXE, APP, JXP, PRG.

If a program was called by another program, then when execution is complete, control is returned to the calling program. If called from the IDE command window, control is passed to the command window, otherwise control is passed back to the operating system.

JAXBase looks for the cProcedure in the currently executing program. If the procedure is not located there, JAXBase then looks for the procedure in the procedure files opened with SET PROCEDURE. You can include the IN ProgramName2 clause to tell JAXBase to look for the procedure in a file you specify.

The parameters listed can be expressions, memory variables, literals, fields, or user-defined functions. By default, arrays and objects are sent by reference. You can pass an array or object by value by placing it in parentheses. An array element for a parameter is passed by value unless the element contains an object, which is passed by reference.

See Also

CANCEL Command
COMPILE Command
APARAMETERS Command
LPARAMETERS Command
PARAMETERS Command
QUIT Command
RETURN Command

DRAFT

DO FORM

DO FORM cForm [NAME cVar [PRIVATE|LOCAL]] [WITH eParameterList] [TO cVar] [NOSHOW]

Status

Slated for Version 1.0

Remarks

Loads and displays a form.

Parameters

cForm

Specifies the name of the form or form set to run.

NAME cVar [PRIVATE]

Specifies a variable or array element with which you can reference the form or form set. associated with the form. The NAME variable is created as a global variable unless PRIVATE is issued.

WITH eParameterList

Specifies the parameters passed to the form or form set. If a form set is run, the parameters are passed to the form set's Init method if the form set's WindowType property is set to ModeLess (0) or Modal (1). The parameters are passed to the Load method if the form set's WindowType property is set to Read (2) or ReadModal (3).

TO cVar

Specifies a variable to hold a value returned from the form. If the variable doesn't exist, JAXBase automatically creates it. Use the RETURN command in the Unload event procedure of the form to specify the return value. If you do not include a return value, the default value of true (.T.) is returned. If you use TO, the WindowType property of the form must be set to 1 (Modal). If the form Init event procedure returns .F., preventing the form from being instantiated, the Unload event procedure will not return a value to VarName.

NOSHOW

Specifies that the form's Show method isn't called when the form is run. When you include NOSHOW and run the form, the form is not visible until the form's Visible property is set to true (.T.) or the form's Show method is called.

Example

```

* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

DO FORM TEST1
RETURN

DEFINE CLASS TEST1 AS FORM
  .Caption="Empty Form - Click close icon to exit -->"
  .HEIGHT=200
  .LEFT=100
  .TOP=100
  .WIDTH=300
  .WindowType=1&& MODAL
ENDDEFINE

```

Note

DO FORM executes the Show method for the form or form set.

If you specify a NAME variable that doesn't exist, JAXBase creates it. If you specify an array element, the array must exist before you issue DO FORM. If the variable or array element you specify already exists, its contents of that element are overwritten.

If you omit the NAME clause, JAXBase creates an object type variable with the same name as the form or form set file. Include PRIVATE to set the scope of the format as a private variable. The form is released when the variable loses scope. If you don't include LINKED, the variable is set as a global variable and will stay in memory until the form is closed, the variable is released, or the program ends.

If the form property RELEASEONEXIT=.F. then the form object will not be released when the form is closed.

DO WHILE/ENDDO

```
DO WHILE lExpression
  Commands
  [LOOP]
  [EXIT]
ENDDO
```

Parameters

lExpression

Specifies a logical expression whose value determines whether the commands between DO WHILE and ENDDO are executed. If lExpression evaluates to true (.T.), the set of commands are executed, otherwise control is passed to the first command following the ENDDO.

Commands

Specifies the set of JAXBase commands to be executed if lExpression evaluates to true (.T.).

LOOP

Returns program control directly back to DO WHILE. The LOOP command can be placed anywhere between DO WHILE and ENDDO.

EXIT

Transfers program control from within the DO WHILE loop to the first command following ENDDO. EXIT can be located anywhere between DO WHILE and ENDDO.

ENDDO

Indicates the end of the DO WHILE and transfers control back to the DO WHILE command.

Example

```
* Demonstrate DO WHILE/ENDDO
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
USE (_JAXFolder+"data\polldata")

LOCATE FOR Precinct=203
DO WHILE Precinct=203 AND NOT EOF()
  ? Name,Age,Party
  SKIP
ENDDO
```

Note

Commands between DO WHILE and ENDDO are executed for as long as the logical expression lExpression remains true (.T.). Each DO WHILE statement must have a corresponding ENDDO statement.

Comments can be placed after DO WHILE and ENDDO on the same line. The comments are ignored during program compilation and execution.

See Also

IF/ELSEIF/ELSE/ENDIF

DO CASE/CASE/OTHERWISE/ENDCASE

DO/UNTIL

FOR/ENDFOR

FOREACH/ENDFOR

DO CASE/CASE/OTHERWISE/ENDCASE

```
DO CASE
  CASE lExpression1
    [Commands]
  CASE lExpression2
    [Commands]]
  ...
  CASE lExpressionN
    [Commands]]
  OTHERWISE
    [Commands]]
ENDCASE
```

Remarks

Executes the first set of commands whose conditional expression evaluates to true (.T.).

Parameters

CASE lExpression1 Commands...

When the first true (.T.) CASE expression is encountered, the set of commands following it is executed. Execution of the set of commands continues until the next CASE or ENDCASE is reached.

Execution then resumes with the first command following ENDCASE. If a CASE expression is false (.F.), the set of commands following it up to the next CASE clause is ignored. Only one set of commands is executed. These are the first commands whose CASE expression evaluates to true (.T.).

Any succeeding true (.T.) CASE expressions are ignored.

OTHERWISE Commands

If all of the CASE expressions evaluate to false (.F.), OTHERWISE determines if an additional set of commands is executed.

If you include OTHERWISE, the commands following OTHERWISE are executed and execution skips to the first command following ENDCASE.

If you omit OTHERWISE, execution skips to the first command following ENDCASE.

Note

DO CASE is used to execute a set of JAXBase commands based on the value of a logical expression. When DO CASE is executed, each logical expressions is evaluated and the first one that evaluates to true (.T.) will have its command block executed.

Comments can be placed after DO CASE and ENDCASE on the same line. The comments are ignored during program compilation and execution.

Example

```
CLEAR ALL
CLOSE ALL

STORE CMONTH(DATE( )) TO month  && The month today

DO CASE  && Begins loop

CASE INLIST(month,'January','February','March')
    STORE 'First Quarter Earnings' TO rpt_title

CASE INLIST(month,'April','May','June')
    STORE 'Second Quarter Earnings' TO rpt_title

CASE INLIST(month,'July','August','September')
    STORE 'Third Quarter Earnings' TO rpt_title

OTHERWISE
    STORE 'Fourth Quarter Earnings' TO rpt_title
ENDCASE  && Ends loop

WAIT WINDOW rpt_title NOWAIT
```

See Also

[IF/ENDIF Command](#)
[FOR/ENDFOR Command](#)
[FOREACH/ENDFOR Command](#)
[DO/UNTIL Command](#)
[DO WHILE/ENDDO Command](#)

DO/UNTIL

```
DO
  Commands
  [EXIT]
  [LOOP]
UNTIL lExpression
```

Remarks

Executes a block of code at least once and continues to execute the block of code until the lExpression resolves as True (.T.).

Parameters

lExpression

Specifies a logical expression whose value determines whether the commands between DO and UNTIL are executed. As long as lExpression evaluates to true (.T.), the set of commands are executed.

Commands

Specifies the set of commands to be executed as long as lExpression evaluates to true (.T.).

LOOP

Sends program control directly to UNTIL. LOOP can be placed anywhere between DO and UNTIL.

EXIT

Transfers program control from within the DO loop to the first command following UNTIL. EXIT can be placed anywhere between DO and UNTIL.

Example

In the following example, the program reads in a text file and prints out each line until a period is found. A DO/UNTIL is most useful when you need to perform a process at least once, or where you may need to duplicate code outside of a while loop to finish processing left over information.

```
CLEAR ALL
CLOSE ALL

I=1
A=FILETOSTR( "test.txt" )
DO
  B=MLINE(A,I)
  ? B
UNTIL "." $ B
```

Note

The code between DO and UNTIL will be executed once and then the UNTIL is evaluated to determine if control is passed back to the DO command.

See Also

DO WHILE/ENDDO Command
FOR/ENDFOR Command
FOREACH/ENDFOR Command
SCAN/ENDSCAN Command

DRAFT

DOEVENTS

DOEVENTS

Status

Slated for after Version 1.0

Remarks

Causes the GUI to refresh.

Note

When changes to a visual object occur in rapid succession, adding DOEVENTS to the code forces the visual object to be updated.

See Also

CREATE FORM Command
CREATEOBJECT() Function

DROP TABLE

```
DROP cTable [FROM cDbName]
```

Status

Slated for Version 1.0

Remarks

Drops a table from a SQL database

Parameters

cTable

Character literal indicating the name of the SQL table to drop.

FROM cDbName

Character literal indicating the name of SQL database that the table is being dropped from.

Note

If the FROM clause is omitted, the currently open database.

When using FROM cDbName, you must have the database open either using the OPEN DATABASE command or creating a connection using the SQL class.

See Also

[CREATE TABLE Command](#)
[OPEN DATABASE Command](#)
[SQL Class](#)

EDIT

EDIT [TO ovar] [NOSHOW]

Status

Slated for after Version 1.0

Remarks

Brings up the Edit form for the table or cursor open in the current work area.

Parameters

TO oVar

You can assign the edit form to a variable name in order to open up more than one edit form.

NOSHOW

Creates the edit form but does not show it.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
EDIT
```

Note

When the EDIT form is created, the current various environment variables are assigned and many are read-only. See the EDIT class for more information.

EDIT starts with the current record. If you reach the end of the file, or click on NEW then a blank record is appended. If hit the ESC key, the current records changes are discarded.

If you created a blank record or made alterations to the current record and click on the CLOSE button, the record is saved.

If you edit a record and move to another row, the changes are saved.

Errors generated by the EDIT form will be displayed in a dialog, allowing you to fix them before continuing.

The EDIT window has properties that can be altered by code.

See Also

APPEND Command
BROWSE Command

DRAFT

ERROR

ERROR [nExpression]

Status

Slated for Version 1.0

Remarks

Forces the generation of the specified JAXBase error.

Parameters

nExpression

Numeric expression used to generate an error. Default is 9000.

Example

```
* Demonstrate ERROR command
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

ON ERROR ? "Error",ERROR()
ERROR 10
```

Note

If an error handler is currently set, JAXBase will attempt to execute it.

See Also

ON ERROR

Error Method

EXIT

EXIT

Remarks

The EXIT command transfers control to the command immediately after the current FOR, WHILE, SCAN, DO UNTIL, or TRY/CATCH/ENDTRY/FINALLY structures.

Example

```
* Demonstrate EXIT
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

FOR i = 1 TO 17
  IF i < 10
    ? i
  ELSE
    ? "EXIT!"
    EXIT
  ENDIF
ENDFOR
```

See Also

DO WHILE/ENDDO Command
FOR EACH/ENDFOR Command
FOR/ENDFOR Command
SCAN/ENDSCAN Command
TRY/CATCH/ENDTRY/FINALLY Command

EXTERNAL

EXTERNAL ARRAY cArrayList
EXTERNAL FILE | PROGRAM | CLASS | CLASSLIB | FORM | LABEL | QUERY cName

Status

Slated for Version 2.0

Remarks

During compilation, JAXBase looks for references to arrays and files in the code. The EXTRNAL command overrides missing references to various files or arrays in the program where they are referenced.

Parameters

ARRAY cArrayList

List of comma delimited character literals indicating arrays that are initialized externally to the current program, procedure, method, or event.

FILE | PROGRAM | CLASS | CLASSLIB | FORM | LABEL | QUERY cName

Indicates the type of eternal file and name.

Example

```
* Demonstrate EXTERNAL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

EXTERNAL FORM (_JAXFolder+"samples\testform.scx")
DO FORM TEST
```

Note

If a listed parameter name matches a the name of an unknown array, it is assumed that the parameter will be an array.

See Also

BUILD APPLICATION Command
 BUILD EXECUTABLE Command
 BUILD PROJECT Command

FOR/ENDFOR

```
FOR cVar = nStart TO nEnd [STEP nStep]
    [EXIT]
    [LOOP]
ENDFOR
```

Remarks

The FOR/ENDFOR loop allows you to execute the same code until the numeric end expression is reached.

Parameters

cVar

Character literal of the variable to create or overwrite for incrementing through the FOR loop

nStart

Numeric expression of the starting value.

nEnd

Numeric expression of the ending value.

nStep

Numeric expression of the incremental value. A positive value increments nStart up while a negative value decrements nStart down.

Example

```
* Demonstrate FOR LOOP
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

FOR I = 10 to 1 STEP -1
    IF I=2
        LOOP
    ENDIF

    ? I
ENDFOR
```

Note

When the for loop is first executed, the numeric starting expression is placed into the variable and the numeric end expression along with the numeric step value (if missing the value is 1) are recorded.

If the step value is positive, then as long as the variable's value is not greater than the end expression's original value, looping will continue.

If the step value is negative, then as long as the variable's value is not less than the end expression's original value, looping will continue.

If the step value is zero (0) then looping will continue unless an EXIT command is encountered.

If an EXIT command is executed, the loop is exited and the code following the ENFOR is executed.

If a LOOP command is executed, control is passed to the ENDFOR.

When the ENFOR is executed, control is passed to the FOR statement. The variable will be incremented by the numeric step value. If the variable's value is out of range for the numeric end expression, then control is passed to the command following the ENDFOR, otherwise the loop's code is executed again.

See Also

DO/UNTIL
DO WHILE/ENDDO
FOREACH/ENDFOR
SCAN/ENDSCAN

FOREACH/ENDFOR

Status

Slated for Version 2.0

DRAFT

FUNCTION

FUNCTION cName

Remarks

Marks the start of a user defined function.

Parameters

cName

Character literal name of the function it will be referenced as.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
DO TEST1 WITH 9
RETURN
```

```
FUNCTION TEST1
PARAMETERS nVal
? nVal
RETURN
```

Note

FUNCTION is included for compatibility with other dialects. FUNCTION is equivalent to PROCEDURE in JAXBase.

See Also

PROCEDURE Command
Methods and Events

GATHER

```
GATHER FROM ARRAY cArray | MEMVAR | NAME cObject | JSON cExpression  
[FIELDS FieldList | FIELDS LIKE Skeleton | FIELDS EXCEPT Skeleton]  
[MEMO][BLANK]
```

Status

Slated for Version 1.0

Remarks

Moves information from memory to the current record.

Parameters

FROM ARRAY cArrayName

If a FIELDS expression exists, works from Column 1 to column x filling in the fields of the record.

If there are no FIELDS expression, starts with the first field and moves towards the last field, filling them in with information from the columns of the array, except for MEMO, GENERAL, and BLOB fields are always skipped.

MEMVAR

Memory variables that match the fields in the table are used to update the current record.

NAME cObject

The object's properties are used to find matching field names to update.

JSON cExpression

Places matching field information from the JSON object in to the record.

FIELDS FieldList

Specifies the fields to populate from the array.

FIELDS LIKE

Specifies a field skeleton where matching fields are populated from the array.

FIELDS EXCEPT

Specifies fields to skip when populating from an array

MEMO

Updates memo, general, and blob fields if there are matching memory references. MEMO will not work with FROM ARRAY.

BLANK

If there is a FIELDS expression, blanks out all values for the matching fields. Otherwise blanks out the fields with matching names in memory.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

cInfo= "{Name: "JOHN SMITH", Age: 30}, {Name: JANE DOE", Age: 28}"

CREATE TABLE TEST1 (Name C(20), Age I)
GATHER FROM JSON cInfo
BROWSE
```

Note

You must have ARRAY, MEMVAR, NAME, or JSON after FROM to specify how to save the requested fields.

When gathering from an array, If there are too few columns, the rest of the fields are ignored, and if there are too many columns, the rest of the columns are ignored.

If there is more than one row in the array, the extra rows are ignored.

If a JSON string has more than one entry, only the first entry is used to update the current record.

See Also

[REPLACE Command](#)
[SCATTER Command](#)

GETEXPR

Status

Slated for Version 2.0

DRAFT

GOTO

GOTO TOP | BOTTOM | nExpression [IN nWorkArea | cAlias]

Remarks

Moves the table or cursor record pointer to the indicated physical record.

Parameters

TOP

Literal indicating the first record of the table or cursor.

BOTTOM

Literal indicating the last record of the table or cursor.

nExpression

IN nWorkArea | cAlias

Example

* Demonstrate GOTO

CLEAR ALL

CLOSE ALL

SET CONSOLE ON

ACTIVATE CONSOLE

CLEAR

USE (_JAXFolder+"data\polldata")

GOTO 15

BROWSE

Note

If nExpression is less than 1 or greater than the number of records in the table, an error is generated.

Issuing TOP moves the record pointer to the top of the table.

Issuing BOTTOM moves the record pointer to the bottom of the table.

If there is an active index, TOP and BOTTOM move to the respective top and bottom of the sorted table.

If there is no table open in the indicated work area or the alias is not found, an error is generated.

See Also

SET INDEX Command

SKIP Command

USE Command

DRAFT

HELP

Status

Slated for Version 2.0

DRAFT

IF/ELSEIF/ELSE/ENDIF

```
IF lExpression
    [Command block]
[ELSEIF lExpression
    Command block]
[ELSE
    Command block]
ENDIF
```

Remarks

The IF/ELSEIF/ELSE/ENDIF structure allows for rapid decision making on what code to execute.

Parameters

lExpression

Expression to check. If it evaluates to true (.T.) then the command block for the IF is executed.

ELSEIF lExpression

Another IF expression to check. If it evaluates to true (.T.) then the command block for that ELSEIF is executed.

ELSE

If the IF expression and any ELSEIF expressions evaluate to false (.F.), then the ELSE block, if it exists, is executed.

Command Block

One or more JAXBase commands to execute.

Example

```
* Demonstrate IF statement
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

IF 1=2
    ? "That's not right!"
ELSEIF 2=3
    ? "That's still not right"
ELSE
    ? "Everything is great!"
ENDIF
```

Note

ELSEIF allows you to change:

```
IF Expression
ELSE
    IF Expression
    ELSE
        IF Expression
        ENDIF
    ENDIF
ELSE
ENDIF
```

To something that looks like:

```
IF lExpression
ELSEIF lExpression
ELSEIF lExpression
ELSE
ENDIF
```

See Also

CASE/OTHERWISE/ENDCASE
DO/WHILE
DO/UNTIL
FOR/ENDFOR
FOREACH/ENDFOR

IMPORT

```
IMPORT FROM FileName
  [TYPE] XLS | XL5 | XLSX | CALC [SHEET cSheetName]
  [FOR lExpr] [AS nCodePage]
```

Status

Slated for Version 2.0

DRAFT

INDEX

```
INDEX ON eExpression TO cFile
[COLLATE cCollateSequence] [FOR lExpression]
[ASCENDING | DESCENDING] [UNIQUE | CANDIDATE] [NOCASE]
```

Status

If CDX files are implemented in (Version 2.0 or later)

```
INDEX ON eExpression TAG cTag [BINARY]
[COLLATE cCollateSequence] [OF CDXFileName] [FOR lExpression]
[ASCENDING | DESCENDING] [UNIQUE | CANDIDATE] [NOCASE]
```

Remarks

Creates an index file (IDX)

Parameters

ON eExpression

Creates an index based on the given expression.

TO cFile

Saves the index using the character literal expression as the file stem.

COLLATE cCollateSequence (Version 2.0)

Character literal expression indicating

FOR lExpression

ASCENDING | DESCENDING

UNIQUE | CANDIDATE

The UNIQUE keyword will create an index that only inserts the first record that matches the key, while the CANDIDATE keyword prevents duplicate keys from being inserted into the table. If a IDX index is used, it must be open whenever a record is added or a key field is altered, otherwise the index will not correctly deal with new or changed keys.

When a PACK statement is run against a table, all open indexes are automatically reindexed. If a record is marked for deletion, a SEARCH will always locate it if matched, but a LOCATE statement will not.

NOCASE

Traditional XBase languages have typically been incompatible with how a SQL database works in a few respects when dealing with indexes. One major incompatibility is that a XBase index tends to be case sensitive while SQL indexes tend to be case insensitive for character data. The NOCASE parameter tells JAXBase to build an index that is case insensitive. If the eExpression is not character based (CHAR, VARCHAR, VARBINARY, VARCHAR (Binary)) then an error is raised.

When a SEEK is performed on an index that has been created with the NOCASE parameter, the key search is done on a case insensitive basis.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\phonebook")
INDEX ON name to phonebook_name_nocase NOCASE
? SEEK("smith"), LastName && Displays ".T. Smith"
```

Note

When an index is created, any other open indexes remain open, but after the indexing is completed, the newly created index is now the controlling index.

When a simple index (IDX File) command is executed, if the IDXFileName already exists, it will overwrite the old file. If you run the index command against a CDX file with a similar name, it too will overwrite the old named index.

You cannot index a memo, general, blob, or JASON field unless you take some sort of subset of the data from the field, such as the following being used against a memo field named TextData.

```
INDEX ON MLINE(TextData,1) TO TextData_Line1
INDEX ON JSON(TextData,"Name") TO TextData_Name
```

All index expressions are limited to 254 characters.

See Also

- CANDIDATE() Function
- UNIQUE() Function
- COPY INDEXES Command
- DESCENDING() Function
- ASSQL()
- FOR() Function
- INDEXSEEK() Function
- KEY() Function
- MDX() Function
- NDX() Function
- ORDER() Function
- SET INDEX Command
- SORT Command

INSERT

```
INSERT INTO cTable (FieldName1[, FieldName2, ...]) VALUES (eExpression1[, eExpression2, ...])
```

Slated for Version 2.0

```
INSERT INTO cTable [(FieldName1[, FieldName2, ...])]  
    SELECT SELECTClauses [UNION UnionClause SELECT SELECTClauses ...]
```

Status

Slated for after Version 1.0

Remarks

Insert a record into a DBF table or SQL table.

Parameters

cTable

Literal character name of DBF or SQL table to insert a record.

FieldName1[, FieldName2, ...]

List of character literals specifying the field names to insert into.

eExpression1[, eExpression2...]

List of expressions specifying the values to place int the fields.

SELECT SELECTClauses [UNION UnionClause SELECT SELECTClauses ...]

Sub queries will be explained in greater detail in an upcoming release.

Example

```
* Demonstrate INSERT  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
CREATE TABLE MyTable (FirstName C(30), LastName C(30), BirthDate D)  
INSERT INTO MyTable (FirstName, LastName, BirthDate) VALUES ("JOHN", "SMITH", {2001/07/15})  
INSERT INTO MyTable (FirstName, LastName, BirthDate) VALUES ("JANE", "SMITH", {2004/01/27})  
INSERT INTO MyTable (FirstName, LastName, BirthDate) VALUES ("JACK", "SMITH", {2025/12/30})  
BROWSE  
  
USE
```

Note

Field and values must match. If there is a difference between the number of each, an error is generated.

Expressions data types must be compatible with field types, otherwise an error is generated.

An inserted record is appended to the end of a table, and therefore acts much like the APPEND Command.

See Also

APPEND Command

GATHER Command

SCATTER Command

DRAFT

KEYBOARD

KEYBOARD *cKeyValue* [PLAIN] [CLEAR]

Status

Slated for Version 1.0

Remarks

Insert keystrokes into the keyboard buffer.

Parameters

cKeyValue

Character expression that will be placed into the keyboard buffer.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
KEYBOARD "DIRECTORY"+CHR(13)
```

Note

KEYBOARD is used to push keystrokes into the keyboard buffer to produce automation.

See Also

MOUSE Command

CLEAR TYPEAHEAD Command

LIST

List commands are exactly like Display commands, but where display commands will pause at the end of every screen of information displayed, list commands will display all the information without stopping until it is done.

See the Display command for more information.

DRAFT

LOCATE

`LOCATE [FOR lExpression1] [Scope] [WHILE lExpression2]`

Remarks

Starting at the top of the file, sequentially searching for the first record that matches the logical expressions. Works only on the table in the current work area.

Parameters

FOR lExpression

Expression is used to locate the record that matches the logical expression. Expressions are case sensitive unless you take steps to make it case insensitive.

SCOPE

Specifies the range of records to use. Only records that fall within the scope are located. Scope clauses: ALL, NEXT nRecords, RECORD nRecordNo, and REST. The default scope is ALL.

WHILE lExpression

LOCATE will continue to search as long as this condition is true.

Example

```
SET TALK OFF
CLOSE DATABASES
USE customers && Open Customer table

STORE 0 TO gnCount

LOCATE FOR ALLTRIM(UPPER(customers.country)) = 'CANADA'

DO WHILE FOUND()
  gnCount = gnCount + 1
  ? company,address,city,country
  CONTINUE
ENDDO

? 'Total companies: '+ LTRIM(STR(gnCount))
```

Note

The table does not require an index. If an index is active, it will search the records in the order indicated by the index.

LOCATE without a FOR expression will go to the first logical record of the table.

If a match is not found, the record position is placed after the end of the table and RECNO() returns the number of records+1, FOUND() returns .F. (false), EOF() returns .T. (true). You can issue a CONTINUE command after a successful LOCATE to attempt to find the next matching record.

LOCATE is specific to the current workarea.

See Also

CONTINUE Command
EOF() Function
FOUND() Function
INDEXSEEK() Function
RECNO() Function
SEEK Command
SEEK() Function

DRAFT

LOCAL

```
LOCAL var1 [,var2 [,var3...]]
```

Remarks

The LOCAL command creates a local variable or array whose scope is the currently executing code module. All variable elements are set to .F. upon initialization.

Parameters

Variable list containing variables or array declarations.

Example

```
LOCAL a,b,c[3]
DISPLAY memory like *
```

```
A      LOC L .F.
B      LOC L .F.
C      LOC A
      (1) L .F.
      (2) L .F.
      (3) L . F.
```

Note

Some xBase languages use the LOCAL command as a flag setting and can have wildcards in the variable name so that if a matching variable is created, it is marked as local. JAXBase expects you to specify the exact variable name.

If the variable is a private variable that was created in the current scope, an error will be thrown.

See Also

PRIVATE
PUBLIC

LOOP

LOOP

Remarks

Transfers control to the end of the FOR, WHILE, SCAN, or UNTIL structure.

Example

```
* Demonstrate LOOP
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

FOR i = 1 TO 17
    IF i = 5
        LOOP
    ENDIF

    ? i
ENDFOR
```

See Also

DO WHILE/ENDDO Command
FOR EACH/ENDFOR Command
FOR/ENDFOR Command
SCAN/ENDSCAN Command
TRY/CATCH/ENDTRY/FINALLY Command

LPARAMETERS

LPARAMETERS var1 [*As Type1*] [,var2 [*As Type2*] ...]

Remarks

Allows information to be received from a calling routine and places those values into the list of LOCAL variables. If the calling routine sends fewer values than the LPARAMETERS statement allows, the remaining variables are set to .F. (false). If the calling routine sends more values than are listed in the LPARAMETERS statement, an error is raised.

Parameters

Var

Variable name to use in this routine

As Type (Slated for Version 1.0)

Allows for strong typing and if the value passed is not of a compatible type or a value is later assigned that is not of a compatible type, an error is raised. By Type only refers to JAXBase field types (B,C,D,F,I,L,N,T) and standard object classes.

Values from General, Blob, Memo, JSON, VarChar (V) and Binary VarChar fields are considered Character (C) types.

Date values can be sent to DateTime and the time value will be set to 00:00:00 while sending a DateTime value to a Date type will have the time value dropped.

Any number type (B,F,I, or N) can be sent to any other number type with the appropriate conversion made if the value is not out of range (such as a Double value sent to an Integer that is outside of the -2147483647 to 2147483647 range would raise an error).

Using the EMPTY class will allow any object to be passed. Attempting to send an array or other data type value to an object type will raise an error.

The ARRAY type will accept any array. If a field type value or object is sent, the results will be that value placed into an array, by value, with 1 element. If an array is sent, it is passed by reference.

Example

```
* Demonstrate LPARAMETER
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET TALK OFF
```

```
DO TESTREF WITH 1,2,3
```

```
RETURN
```

```
PROCEDURE TESTREF
```

```
LPARAMETERS AA1, BB1, CC1
```

```
DO TESTLOCAL
```

```
RETURN
```

```
* Try to print out a variable from the TESTREF procedure
```

```
PROCEDURE TESTLOCAL
```

```
TRY
```

```
? "BB1=",BB1
```

```
CATCH
```

```
?? "BB1 is not available"
```

```
ENDTRY
```

```
RETURN
```

Note

The LPARAMETERS statement creates the local variables from the list based on the information sent by the calling routine. If a value, array or object is sent, the associated parameter variable becomes a copy of that value, array or object.

Passing by Reference may not be available before Version 1.0

By default, DO/WITH passes objects and arrays by reference unless the object is part of an expression or a single array element is passed. All other calls, such as a function call or object call, are passed by value.

See Also

[APARAMETERS Command](#)

[PARAMETERS Command](#)

[DO/WITH Command](#)

[JAXBase Field Types](#)

[UDF Calls](#)

[Object Calls](#)

LPROCEDURE

LPROCEDURE cName

Remarks

Creates a procedure that is local to the object or program file to which it is associated and makes it so that procedure cannot be called by code outside of the object or program file.

Parameters

Character literal used as the procedure name.

Example

```
* -----
* Parent.prg
* -----
DO child.prg

PROCEDURE Test1
? "Test 1 routine executed"
RETURN
ENDPROCEDURE

LPROCEDURE Test2
? "Test 2 routine executed"
RETURN
ENDPROCEDURE

* -----
* Child.prg
* -----
TRY
    Do Test1
CATCH
    ? "Failed to call Test1 in Parent.prg"
ENDTRY

TRY
    Do Test2
CATCH
    ? "Failed to call Test2 in Parent.prg"
ENDTRY
RETURN
```

Note

A LPROCEDURE code block, like PROCEDURE is terminated with ENDPROCEDURE, but JAXBase does not require the ENDPROCEDURE command as it is only included for readability.

See Also

[PROCEDURE Command](#)

[DO Command](#)

[User Defined Functions](#)

MD

MD cDirectory

Remarks

Creates a directory if it does not exist.

Parameters

cDirectory

A literal character expression defining the directory to create, such as C:\TEMP

Example

```
* Demonstration of MD command  
MD C:\TEMP
```

```
cFolder="C:\TEMP\JAXBase\Jobs"  
MD (cFolder)
```

Note

The folder expression must be literal expression, such as `MD C:\TEMP` or an expression inside parenthesis, such as `MD (cFolder)`, otherwise an error will be raised. If the folder already exists, the command continues as normal.

If the folder cannot be created, an error will be generated.

See Also

ADIR() Function

CD Command

DIRECTORY/DIR Commands

MODIFY

Modify Class

Status

Slated for after Version 1.0

Remarks

Brings up the Class Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Command

Status

Slated for Version 1.0

Remarks

Brings up the Program Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify File

Status

Slated for Version 1.0

Remarks

Brings up the File Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Form

Status

Slated for Version 1.0

Remarks

Brings up the Form Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Label

Status

Slated for after Version 1.0

Remarks

Brings up the Label Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Menu

Status

Slated for after Version 1.0

Remarks

Brings up the Menu Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Project

Status

Slated for Version 1.0

Remarks

Brings up the Project Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Query

Status

Slated for after Version 1.0

Remarks

Brings up the Query Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Report

Status

Slated for Version 1.0

Remarks

Brings up the Report Editor application

Note

The project and source code for this editor can be found in the TOOLS folder

Modify Structure

Status

Slated for Version 1.0

Remarks

Brings up the Table Structure editor application

Note

The project and source code for this editor can be found in the TOOLS folder

MOUSE

```
MOUSE [TO xPos, yPos] [MOVE xPixels, yPixels] {CLICK LEFT | MIDDLE | RIGHT}
```

Status

Slated for after Version 1.0

Remarks

The mouse command is used to move the mouse cursor around in order to create automation.

Parameters

TO xPos, yPos

Moves the mouse cursor to the absolute x,y position.

MOVE xPixels, yPixels

Moves the mouse cursor the number of pixels along the x and y axis.

CLICK LEFT | MIDDLE | RIGHT

Simulates the clicking of the Left, Middle, or Right mouse key.

Example

```
* Demonstrate MOUSE command
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
aFRM=CREATEOBJECT( "FORM" )
aFRM.WindowType=1    && Modal
aFRM.LEFT=100
aFRM.TOP=100
aFRM.SHOW
```

```
* Move onto the form then to the top right corner
```

```
MOUSE TO aFRM.LEFT+20, aFRM.TOP+30 TO aFRM.LEFT+aFRM.WIDTH, aFROMTOP
INKEY(3)
```

```
* Move down and left into the close icon and click to close the form
MOUSE MOVE -5,-5 CLICK LEFT
```

Note

Creating automation in a JAX application is a straightforward task using the KEYBOARD and MOUSE commands.

You can use the TO, MOVE, and CLICK parameters in the same statement.

See Also

KEYBOARD Command

ON

ON ESCAPE

ON ESCAPE Command

Status

Slated for Version 1.0

Remarks

Executes a JAXBase Command when the ESC key is pressed.

Parameters

Command

Any valid single-line JAXBase command

Example

```
ON ESCAPE ? "You hit the ESC Key!"
```

See Also

The ON ESCAPE command cannot execute any multi-line command, such as DO WHILE/ENDDO.

The most common command executed by an ON ESCAPE command is DO Program

Issuing ON ESCAPE with no parameters disables the command.

ON KEY LABEL

ON KEY LABEL <key> Command

Status

Slated for Version 1.0

Remarks

Executes a JAXBase Command when the a designated key is pressed.

Parameters

<key>

A key label as described in [Using Keyboard Commands](#)

Command

Any valid single-line JAXBase command

Example

```
ON KEY LABEL ALT+F5 ? "You hit the ALT F5 Key!"
```

See Also

The ON ESCAPE command cannot execute any multi-line command, such as DO WHILE/ENDDO.

The most common command executed by an ON KEY LABEL command is DO Program

Issuing ON KEY LABEL <key> with no command disables that label command.

ON SHUTDOWN

ON SHUTDOWN Command

Status

Slated for Version 1.0

Remarks

Executes a JAXBase Command when a program is being terminated.

Parameters

Command

Any valid single-line JAXBase command

Example

```
ON SHUTDOWN DO CleanUp.prg
```

See Also

The ON SHUTDOWN command cannot execute any multi-line command, such as DO WHILE/ENDDO.

The most common command executed by an ON SHUTDOWN command is DO Program

Issuing ON SHUTDOWN with no parameters disables the command.

OPEN

```
OPEN DATABASE AliasName [USING cExpr] [NOUPDATE]
```

Remarks

Opens a backend SQL database connection for use in JAXBase.

Parameters

AliasName

Indicates that the database should be referenced using the provided alias name. All database and table alias names must be unique.

USING

The connection string expression following the USING command is used to open the SQL connection. If there is no USING command, then the system relies on a previous SET SQLCONNECTION command for connection information, but if SQLCONNECTION information is null or blank, an error is generated.

NOUPDATE

Marks a database connection as read-only and will not allow changes to be sent back through that connection.

Example

```
* OPEN THE DATABASE
USE DATABASE JAXCorp

* GET THE OFFICERS TABLE
JC.EXEC("SELECT * FROM OFFICERS", "results")

* UPDATE THE JOHN SMITH RECORD
IF USED("results")
    SELECT results
    LOCATE FOR NAME="JOHN SMITH"
    IF FOUND()
        REPLACE POSITION WITH "CEO"
    ENDIF

    * UPDATE THE OFFICERS TABLE AND
    UPDATE TABLE
ENDIF

* CLOSE THE DATABASE CONNECTION
CLOSE DATABASE JAXCorp
```

Note

To access the data in a database table, you will first need to use a SELECT statement and place the information into a local table or cursor.

If you save to a table, any changes to the database would need to be done manually standard SQL commands, such as UPDATE, INSERT, or DELETE.

If you save the results of a SELECT query to a cursor, you can use the UPDATE TABLE command to push changes back to the database in bulk. Using the UPDATE TABLE command against cursors created using JOIN, UNION, or sub-queries will raise an error. Only field names that exist in the table will be used in an UPDATE TABLE command. Issuing a REFRESH TABLE command will refresh one or more records from the database table (erasing any changes made to the refreshed records).

You can find several programs demonstrating how to use a SQL backend in the EXAMPLES folder.

See Also

[CREATE DATABASE](#)
[CREATE TABLE](#)
[CREATE VIEW](#)
[DELETE DATABASE](#)
[OPEN DATABASE](#)
[REFRESH TABLE](#)
[UPDATE TABLE](#)
[DATABASE\(\)](#)
[Database Class](#)
[Table Class](#)
[SQL Class](#)

PACK

PACK [MEMO | DBF] [IN nWorkarea | cAlias]

Remarks

Removes all records from a DBF file that are marked for deletion.

Parameters

MEMO | DBF

MEMO only packs the memo file while DBF causes the DBF and MEMO files to be packed and any open indexes to be reindexed.

IN nWorkArea | cAlias

The PACK command will be run against the table in nWorkArea or with the alias name of cAlias

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
CREATE TABLE Test1 (NAME c(20))
INSERT INTO TEST1 (NAME) VALUES ("JOHN SMITH")
INSERT INTO TEST1 (NAME) VALUES ("JANE SMITH")
INSERT INTO TEST1 (NAME) VALUES ("JOHN DOE")
INSERT INTO TEST1 (NAME) VALUES ("JANE DOE")
? RECCOUNT(), "records found"
```

```
GOTO 2
DELETE REST
PACK
? RECCOUNT(), "records found"
```

Note

PACK will not work against a table or cursor that is set as READONLY.

If the MEMO keyword is used against a table that has no Memo file, the command is ignored.

If there is no open table in nWorkArea or the cAlias name is not found, an error is generated.

If work area or alias is omitted, the current work area is used.

See Also

[DELETE Command](#)
[RECALL Command](#)
[REINDEX Command](#)
[SET DELETED Command](#)

PARAMETERS

LPARAMETERS var1 [*As Type1*] [,var2 [*As Type2*] ...]

Remarks

Allows information to be received from a calling routine and places those values into the list of PRIVATE variables. If the calling routine sends fewer values than the PARAMETERS statement allows, the remaining variables are set to .F. (false). If the calling routine sends more values than are listed in the PARAMETERS statement, an error is raised.

Parameters

Var

Variable name to use in this routine

As Type (Slated for Version 1.0)

Allows for strong typing and if the value passed is not of a compatible type or a value is later assigned that is not of a compatible type, an error is raised. By Type only refers to JAXBase field types (B,C,D,F,I,L,N,T) and standard object classes.

Values from General, Blob, Memo, JSON, VarChar (V) and Binary VarChar fields are considered Character (C) types.

Date values can be sent to DateTime and the time value will be set to 00:00:00 while sending a DateTime value to a Date type will have the time value dropped.

Any number type (B,F,I, or N) can be sent to any other number type with the appropriate conversion made if the value is not out of range (such as a Double value sent to an Integer that is outside of the -2147483647 to 2147483647 range would raise an error).

Using the EMPTY class will allow any object to be passed. Attempting to send an array or field type value to an object type will raise an error.

The ARRAY type will accept any array. If a field type value or object is sent, the results will be that value placed into an array, by value, with 1 element. If an array is sent, it is passed by reference.

Example

```

* Demonstrate PARAMETERS and passing by value and reference
CLEAR ALL
CLOSE ALL
SET TALK OFF
PRIVATE A[2], B, C
DO ResetVars

* Passing A and C by reference
DO TESTREF WITH A,B,C
? "Test 1 Results"
? REPLICATE("-",20)
DISPLAY MEMORY EXTENDED
INKEY(5)

* C is passed by value because it is an expression
DO TESTREF WITH A,B,(C)
? "Test 2 Results"
? REPLICATE("-",20)
DISPLAY MEMORY EXTENDED
INKEY(5)

* Only passed by value
=TESTREF(A,B,C)
? "Test 3 Results"
? REPLICATE("-",20)
DISPLAY MEMORY EXTENDED
INKEY(5)

RETURN

PROCEDURE TESTREF
PARAMETERS AA1 AS ARRAY, BB1 AS N, CC1 AS EMPTY
AA[1]=7
BB1=3
CC1.NAME="NewName"

DO TESTLOCAL
RETURN

* Try to print out a variable from the TESTREF procedure
PROCEDURE TESTLOCAL
TRY
    ? "BB1=",BB1
CATCH
    ?? "BB1 is not available"
ENDTRY
RETURN

* Reset the variables for testing
PROCEDURE ResetVars
A[1]=1
A[2]=2
B=7.01
C=CREATEOBJECT("FORM")
RETURN

```

Note

The LPARAMETERS statement creates the local variables from the list based on the information sent by the calling routine. If a value, array or object is sent, the associated parameter variable becomes a copy of that value, array or object.

Passing by Reference may not be available before Version 1.0

By default, DO/WITH passes objects and arrays by reference unless the object is part of an expression or a single array element is passed. All other calls, such as a function call or object call, are passed by value.

See Also

[APARAMETERS Command](#)
[PARAMETERS Command](#)
[DO/WITH Command](#)
[JAXBase Field Types](#)
[UDF Calls](#)
[Object Calls](#)

PLAY

```
PLAY cExpression1 [, cExpression2 ...] [TIME nDelay]
PLAY MACRO cName [TIME nDelay]
```

Status

Slated for after Version 1.0

Remarks

Send keystrokes to the keyboard buffer with optional delays between each expression or keystroke.

Parameters

cExpression1 [, cExpression2 ...]

Comma delimited character expressions containing keys you wish to push to the keyboard buffer.

MACRO cName

Character literal of the name of the keyboard macro to execute.

TIME nDelay

Numeric expression specifying the delay, in milliseconds, between each MACRO or each line terminated with character 13 (carriage return). The carriage return is not sent with the keystrokes.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
PLAY "DIRECTORY" , "{RETURN}" , "CLEAR" , "{RETURN}" TIMEOUT 2000
```

Note

This command is different than other dialects.

{RETURN} simulates the ENTER/RETURN key on the keyboard. For more information, please see the chapter o.

See Also

CLEAR TYPEAHEAD Command

INKEY() Function

KEYBOARD Command

MOUSE Command

PRIVATE

```
PRIVATE var1 [,var2 [,var3...]]
```

Remarks

The PRIVATE command creates a variable or array whose scope is the currently executing code module and any code modules after that point. All variable elements are set to .F. upon initialization

Parameters

Variable list containing variables or array declarations.

Example

```
LOCAL a,b,c[3]
DISPLAY memory like *
```

```
A      LOC L .F.
B      LOC L .F.
C      LOC A
      (1) L .F.
      (2) L .F.
      (3) L . F.
```

Note

Some xBase languages use the LOCAL command as a flag setting and can have wildcards in the variable name so that if a matching variable is created, it is marked as local. JAXBase expects you to specify the exact variable name.

If the variable is a private variable that was created in the current scope, an error will be thrown.

See Also

LOCAL
PUBLIC

PROCEDURE

LPROCEDURE cName

Remarks

Creates a procedure for a program file to which it is associated.

Parameters

Character literal used as the procedure name.

Example

```
* -----
* Parent.prg
* -----  
  
DO child.prg  
  
PROCEDURE Test1  
? "Test 1 routine executed"  
RETURN  
ENDPROCEDURE  
  
LPROCEDURE Test2  
? "Test 2 routine executed"  
RETURN  
ENDPROCEDURE  
  
* -----  
* Child.prg  
* -----  
  
TRY  
    Do Test1  
CATCH  
    ? "Failed to call Test1 in Parent.prg"  
ENDTRY  
  
TRY  
    Do Test2  
CATCH  
    ? "Failed to call Test2 in Parent.prg"  
ENDTRY  
RETURN
```

Note

A PROCEDURE code block, like LPROCEDURE is terminated with ENDPROCEDURE, but JAXBase does not require the ENDPROCEDURE command as it is only included for readability.

See Also

PROCEDURE Command
DO Command
User Defined Functions

PUBLIC

```
PUBLIC var1 [,var2 [,var3..]]
```

Remarks

The PUBLIC command creates a variable or array whose scope is the entire application. All variable elements are set to .F. upon initialization.

Parameters

Variable list containing variables or array declarations.

Example

```
LOCAL a,b,c[3]
DISPLAY memory like *

A      LOC L .F.
B      LOC L .F.
C      LOC A
      (1) L .F.
      (2) L .F.
      (3) L . F.
```

Note

Some xBase languages use the LOCAL command as a flag setting and can have wildcards in the variable name so that if a matching variable is created, it is marked as local. JAXBase expects you to specify the exact variable name.

If the variable is a private variable that was created in the current scope, an error will be thrown.

See Also

LOCAL
PRIVATE

QUIT

Quit [nStatus]

Remarks

Terminates the executing program, and if in the IDE it will cause JAXBase to terminate as well.

Parameters

nStatus

Numeric expression containing the status to return to the operating system.

Example

```
* Demonstrate QUIT
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

QUIT
```

See Also

CANCEL
SUSPEND

RD

RD cDirectory

Remarks

Removes the indicated directory.

Parameters

cDirectory

Character literal indicating the directory to remove.

Note

If any reason the directory cannot be removed, an error is generated.

See Also

ADIR() Function

DIRECTORY/DIR Commands

MD Command

READ EVENTS

DO EVENTS

Status

Slated for Version 1.0

Remarks

Causes JAXBase to start event processing.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

* Call a nonmodal form and wait for EXIT button
DO FORM (_JAXFolder+"examples\nonmodal.scx")
READ EVENTS
```

Note

When READ EVENTS is issued in a program, execution pauses at that point and event processing begins.

Only object methods and events and the code called from them are executed during event processing.

When CLEAR EVENTS is executed, event processing ends and the program continues with the command following READ EVENTS.

Only one READ EVENTS can be active at one time. If another READ EVENTS is encountered during event processing, it is ignored.

See Also

CLEAR Commands
DOEVENTS Command

RECALL

`RECALL [Scope] [FOR lExpression1] [WHILE lExpression2] [IN nWorkArea | cAlias]`

Remarks

Removes the deletion mark from the specified records in the indicated work area or alias.

Parameters

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpression1

Logical expression used to filter what records are used.

WHILE lExpression2

Logical expression limiting the records searched to while the expression is true.

IN nWorkArea | cAlias

Numeric work area literal or character literal for alias name of table to use.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

CREATE TABLE Test1 (Name C(20))
INSERT INTO TEST1 (NAME) VALUES ("JOHN SMITH")
INSERT INTO TEST1 (NAME) VALUES ("JANE SMITH")
INSERT INTO TEST1 (NAME) VALUES ("JOHN DOE")
INSERT INTO TEST1 (NAME) VALUES ("JANE DOE")
```

```
DELETE ALL
COUNT TO A FOR DELETED()
? A, "deleted records"
```

```
GOTO TOP
RECALL
COUNT TO A FOR DELETED()
? A, "deleted records"
```

See Also

COUNT Command
CREATE TABLE Command
DELETE Command
DELETED() Function
INSERT Command

REINDEX

REINDEX

Status

Slated for Version 1.0

Remarks

Causes all open indexes to be recreated for the current work area.

Note

Indexes can become outdated. REINDEX is used to bring them up to date.

If UNIQUE, CANDIDATE, or DESCENDING keywords were used in the original indexing, they will carry over to the new index.

The table must be open in EXCLUSIVE mode in order to issue the REINDEX command.

See Also

INDEX Command

SET EXCLISIVE Command

SET INDEX Command

USE Command

RELEASE

RELEASE Command

RELEASE [cVarList] [ALL]

Status

Slated for Version 1.0

Remarks

Releases the specified variables from memory, freeing up resources.

Parameters

cVarList

Comma delimited character literals specifying the names of variables to release from memory.

ALL

Releases all variables from memory

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
A=5
B=3
C=7
LIST MEMORY
```

```
?
RELEASE C,D
LIST MEMORY
```

Note

If a variable name in the list does not exist, it is ignored.

See Also

LOCAL Command
PRIVATE Command
PUBLIC Command

RELEASE CLASSLIB

RELEASE CLASSLIB [cLibName] [ALL]

Status

Slated for after Version 1.0

Remarks

Removes the indicated class library from memory and closes the VCX.

Parameters

cLibName

Name of class library to release.

ALL

Closes all open class libraries and removes them from memory.

RELEASE PROCEDURE

```
RELEASE PPROCEDURE [ cProc ][ALL]
```

Remarks

Removes the indicated procedure file from memory.

Parameters

cProc

Name of procedure file to release.

ALL

Closes all open procedure files and removes them from memory.

REMOVE

REMOVE CLASS

REMOVE CLASS cClass of cLibrary

Status

Slated for after Version 1.0

Remarks

Removes a class from a class library file (.vcx).

Parameters

cClass

Name of class to remove.

cLibrary

Name of library to remove the class from.

REMOVE TABLE

REMOVE TABLE cTable

Status

Slated for Version 1.0

Remarks

Removes a table from an open database.

Parameters

cTable

Name of table to remove from current database. To remove a table from a different database open in the current data session, use the Database!Table format.

RENAME

RENAME Command

```
RENAME cFileSource TO cFileDestination
```

Status

Slated for Version 1.0

Remarks

Change the name of a file

Parameters

cFileSource

Name of the file to change with extension.

cFileDestination

Name of the new file name with extension.

Note

If the cFileDestination is in a different path location than cFileSource, the file is effectively moved.

RENAME CLASS

RENAME CLASS cClass of cLibrary to cNewClass

Status

Slated for after Version 1.0

Remarks

Changes the name of a class definition in a class library file (.vcx).

Parameters

cClass

Name of class to change.

cLibrary

Class library in which the class is located.

cNewClass

New name of class.

Rename Table

```
RENAME TABLE cTableSource TO cTableDestination
```

Status

Slated for Version 1.0

Remarks

Renames and/or moves a table and memo field file.

Parameters

cTableSource

Character literal specifying the name of the table.

cTableDestination

Character literal specifying the new name of the table.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
CREATE TEST1 (Name C(20), Info M)
USE
```

```
RENAME TEST1 TO TEST1_NEW
USE TEST1_NEW
```

Note

If the destination table name exists, an error is generated.

If the destination table name has a path that is different from the source table, the name is moved to the new path.

If a memo file for the table exists, will also be renamed and/or moved.

REPLACE

REPLACE Command

```
REPLACE cField1 with eExpression1[, cField2 with eExpression2...]
[SCOPE] [FOR lExpression1] [WHILE lExpression2]
[IN nWorkArea | cAlias]
```

Status

Slated for Version 1.0

Remarks

Updates records in a table.

Parameters

cField1 with eExpression1[, cField2 with eExpression2...]

Specifies the field name and the expression to update its value with.

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression limiting the records used in AVERAGE while the expression is true.

IN nWorkArea | cAlias

Specifies the work area or table alias the replace statement affects

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
CREATE TABLE TEST1 (Name C(20), Age I)
```

```
APPEND BLANK
```

```
REPLACE Name WITH 'John Smith',
          Age   WITH 30
```

```
APPEND BLANK
```

```
REPLACE Name WITH 'John Dpe',
          Age   WITH 33
```

Note

The data type of the expression must be compatible with the data type of the field.

If there is no table open in nWork area, or cAlias is not found, an error will be generated.

See Also

APPEND Command

GATHER Command

INSERT Command

SCATTER Command

DRAFT

REPLACE FROM ARRAY

```
REPLACE FROM ArrayName [FIELDS fieldlist] [SCOPE] [FOR lExpression1] [WHILE lExpression2]
```

Status

Slated for Version 1.0

Remarks

Updates the fields from an array

Parameters

FIELDS FieldList

An optional list of fields to update using the contents of the array.

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression limiting the records used in AVERAGE while the expression is true.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
DIMENSION aRec[3]
aRec[1]='John Smith'
aRec[2]=30
aRec[3]='R'
```

```
CREATE TABLE TEST1 (Name C(20), Age I)
APPEND BLANK
REPLACE FROM ARRAY
```

Note

If the array has more elements than the list of fields to replace, the extra elements are ignored.

If the number of fields to replace is greater than the number of elements in the array, the extra fields are not updated.

If an array element data type is not compatible with the data type of the corresponding field, an error is generated.

See Also

GATHER Command
INSERT Command
SCATTER Command

DRAFT

RESTORE

RESTORE FROM

RESTORE FROM cFile [ADDITIVE]

Status

Slated for after Version 1.0

Remarks

Retrieves variables and arrays saved in a variable file and places them in memory.

Parameters

cFile

Name of variable file name to restore to memory.

ADDITIVE

If present, the variables in the file are added to what is already in memory.

Example

* Demonstrate RESTORE FROM

CLEAR ALL

CLOSE ALL

SET CONSOLE ON

ACTIVATE CONSOLE

CLEAR

A=10

SAVE TO TEST1

A=1

LIST MEMORY

RESTORE FROM TEST1

LIST MEMORY

Note

If a variable being restored has the same name and scope as one that exists, it overwrites it.

Public and private variables in the file are restored as private variables in a program.

Public variables and private variables are restored as public variables if restored in the IDE command window.

Local variables are always restored as local variables.

If ADDITIVE is omitted, all variables in memory are removed before the variable file is restored.

See Also

DIMENSION Command
LOCAL Command
PUBLIC Command
PRIVATE Command
SAVE TO Command

DRAFT

RESTORE MACROS

RESTORE MACROS FROM cFile

Status

Slated for after Version 1.0

Remarks

Restores all macros from a macro file.

Parameters

cFile

Name of macro file used to restore saved macros.

Example

```
* Demonstrate RESTORE MACROS
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

ON KEY LABEL F5 ? "HELLO"
SAVE MACROS TO lcMacro
ON KEY LABEL F5
RESTORE MACROS FROM lcMacro
KEYBOARD "{F5}"
```

Note

If the file extension is not supplied, .fky is assumed.

If the file does not contain valid macro definitions, an error is generated.

Only the macro definitions found in the file will be altered.

See Also

CLEAR Commands
KEYBOARD Command
ON KEY LABEL Command
PLAY Command
RESTORE MACROS Command
Using the Keyboard Commands

RESUME

RESUME

Status

Slated for after Version 1.0

Remarks

Resumes execution after a SUSPEND.

Note

A suspended program will RESUME where execution was suspended.

SUSPEND and RESUME are valuable debugging tools that allow you to create break points in a program running in the IDE.

SUSPEND and RESUME are ignored when running outside of the IDE.

See Also

CANCEL Command

SUSPEND Command

RETRY

RETRY

Status

Slated for after Version 1.0

Remarks

Returns control to the calling program from an error routine.

Parameters

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Note

Unlike RETURN which returns control to statement after the call to the procedure, RETRY returns control to where the line that tripped the error handler to try processing that line again.

RETRY is very useful for creating error handlers for routines that lock files or records.

See Also

ON ERROR Command
RETURN Command
Error Method

RETURN

```
RETURN [eExpression]
```

Remarks

Returns a value to the calling statement.

Parameters

eExpression

Expression to return to the calling statement.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? TEST1(1,2)
RETURN
```

```
PROCEDURE TEST1
PARAMETERS A,B
RETURN A+B
```

Note

If there is no eExpression to return, the true (.T.) value is returned by default.

See Also

DO Command
User Defined Functions

SAVE

SAVE CLASS Command

SAVE CLASS ClassName AS FileName [OVERWRITE]

Status

Slated for after Version 1.0

Remarks

Saves a class definition in a VCX file as a DEF source code file.

Parameters

ClassName

Character literal of the class name to save.

FileName

Character literal indicating file name to save as.

OVERWRITE

If the file exists, overwrite it.

Note

ClassName refers to a class in the current open class library. If it is not found, then an error is generated.

If FileName exists and OVERWRITE is not specified, an error will be generated.

See Also

ACLASS() Function
ADD CLASS Command
AMEMBERS() Function
CREATE CLASS Command
CREATE CLASSLIB Command
DEFINE CLASS Command
MODIFY CLASS Command
RELEASE CLASSLIB Command
REMOVE CLASS Command
RENAME CLASS Command
SET CLASSLIB Command

SAVE MACROS

SAVE MACROS TO cFile

Status

Slated for after Version 1.0

Remarks

Saves all ON KEY LABEL macros to the specified file.

Parameters

cFile

Character literal name file that the macros will be saved.

Example

```
* Demonstrate SAVE MACROS
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

ON KEY LABEL F5 ? "HELLO"
SAVE MACROS TO lcMacro
ON KEY LABEL F5
RESTORE MACROS FROM lcMacro
KEYBOARD "{F5}"
```

Note

If the file extension is not supplied, .fky is used.

If a path is not supplied, the default path is used.

See Also

CLEAR Commands
KEYBOARD Command
ON KEY LABEL Command
PLAY Command
RESTORE MACROS Command
Using the Keyboard Commands

SAVE TO

SAVE TO cFile ALL LIKE Skeleton | ALL EXCEPT Skeleton

Status

Slated for after Version 1.0

Remarks

Stores current variables and arrays to a variable file (.mem).

Parameters

cFile

Character literal indicating name of file to save.

ALL LIKE Skeleton

Specifies that all variables and arrays that match Skeleton will be saved.

ALL EXCEPT Skeleton

Specifies that all variables and arrays except those that match Skeleton will be saved.

Example

```
* Demonstrate SAVE TO  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
A=10  
SAVE TO TEST1  
A=1  
LIST MEMORY  
RESTORE FROM TEST1  
LIST MEMORY
```

Note

Use the RESTORE FROM command to place saved variables back into memory.

You cannot save objects to a variable file.

See Also

LOCAL Command
PRIVATE Command
PUBLIC Command
RESTORE FROM Command

SCAN / ENDSCAN

```
SCAN [Scope] [FOR lExpression1] [WHILE lExpression2]
  [Commands]
  [LOOP]
  [EXIT]
ENDSCAN
```

Remarks

The SCAN command starts at the top of the table in the current work area and executes the commands between it and the ENDSCAN, as long as the FOR and WHILE expressions, if present, evaluate to True (.T.). When the ENDSCAN is reached, the record pointer advances one record and as long as the FOR and WHILE statements allow and the end of the file is not reached, the process continues to loop through the commands.

Parameters

Scope

Specifies a range of records to be scanned and only the records within the range are included. The valid scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, REST, and TOP nRecords. The Scope allows you to limit the number of records processed.

FOR

A logical expression that must be evaluated to True (.T.) for the current record to be processed by the block of code. If the FOR command is not present, it is assumed to be True (.T.).

WHILE

A logical expression that must be evaluated to True (.T.) for processing to continue. If the expression evaluates to False (.F.) then control passes to the command after the ENDSCAN.

Example

```
* Demonstrate CALCULATE
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
USE (_JAXFolder+"data\polldata")

a=0
SCAN FOR PRECINCT=203
  a=a+1
ENDSCAN

? a, "records counted"
```

Note

Both the FOR and WHILE commands are evaluated each time the SCAN command is processed.

ALL is the default SCOPE value one is not provided.

When the ENDSCAN command is executed, the table that was current when entering the SCAN/ENDSCAN loop is reselected, and the record pointer advanced, before passing control back to the SCAN command.

See Also

[DO CASE/ENDCASE Command](#)

[DO/UNTIL Command](#)

[DO WHILE/ENDDO Command](#)

[FOR EACH/ENDFOR Command](#)

[FOR/ENDFOR Command](#)

[Scope Clauses](#)

DRAFT

SCATTER

```
SCATTER TO ARRAY ArrayName | MEMVAR | NAME ObjectName | JSON varName  
[FIELDS FieldNameList | FIELDS LIKE Skeleton | FIELDS EXCEPT Skeleton]  
[ADDITIVE] | [MEMO] | [BLANK]
```

Status

Slated for Version 1.0

Remarks

Moves information from the current record to memory.

Parameters

TO ARRAY cArrayName

Copies specified fields to an array. If the ADDITIVE keyword exists, an error is generated.

MEMVAR

Memory variables are created to hold the fields in the current record. If the memory variables already exist, they are overwritten.

NAME cObject

The current fields are moved to an object based on the EMPTY class.

JSON cExpression

Places field information into a JSON string.

FIELDS FieldList

Specifies the fields to populate from the array.

FIELDS LIKE

Specifies a field skeleton where matching fields are populated from the array.

FIELDS EXCEPT

Specifies fields to skip when populating from an array

MEMO

Memo, general, and blob fields are usually ignored with SCATTER unless the MEMO keyword is included. Arrays cannot be used with the MEMO keyword.

BLANK

Creates or updates the memory objects with empty values instead of field values.

ADDITIVE

Adds the fields to the memory objects. If the memory variable or object exists, it is overwritten. If it does not exist, it is created. You cannot use ADDITIVE with arrays.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
SCATTER NAME oInfo
SCATTER JSON cInfo

DISPLAY MEMORY LIKE ?Info
```

Note

You must have ARRAY, MEMVAR, NAME, or JSON after TO to specify where to save the requested fields.

When scattering to an array, if the ARRAY does not exist, it is created with the corresponding columns. If the array name does exist, it is overwritten.

You can only SCATTER one row at a time.

See Also

[REPLACE Command](#)
[GATHER Command](#)

SEEK Command

```
SEEK eExpression
  [ORDER nIndexNumber | cIndexName] [ASCENDING | DESCENDING]
  [IN nWorkArea | cAlias]
```

Status

Slated for Version 1.0

Remarks

SEEK uses an index to search for the expression in the indicated work area or table alias.

Parameters

eExpression

Expression matching the controlling index key.

ORDER nIndexNumber | cIndexName

Numeric or character literal indicating the index number from the current index list or index name.

ASCENDING | DESCENDING

ASCENDING is the default sort order, DESCENDING sets the sort order in reverse.

IN nWorkArea | cAlias

Numeric work area literal or character literal for alias name of table to use.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
SEEK 203 ORDER PoolData_Precinct
BROWSE
```

Note

If the indicated cIndexName is not in the index list, an attempt will be made to find it in the JAXBase path list and open it. Failing that, an error is generated.

If the ORDER parameter is omitted, the current controlling index is used.

If the IN parameter is omitted, the current work area is used.

If there is no table open in the specified work area or the alias is not found, an error is generated.

See Also

INDEX Command
SEEK() Function
USE Command

DRAFT

SELECT Command

```
SELECT nExpr|cExpr
SELECT DATASESSION nExpr
```

Remarks

Selects the requested workarea or data session. Both work areas and data sessions are represented with a positive integer value.

Parameters

DATASESSION nExpr

When included, directs the selection of a data session by numeric ID. If omitted, the numeric work area ID in the current data session is selected.

nExpr

Selects the work area based on the positive integer number. If using an expression like SELECT 10 then work area 10 is selected, but if you use a math expression or variable, then you need to surround it with parenthesis indicating it is an expression to be computed.

* Two ways to select work area 4
SELECT 4 && Integer literal

A=4
SELECT (a) && Integer expression

cExpr

Specifies a work area alias if using the letters A through J, which represent work areas 1 through 10, otherwise it looks for an open table with the specified alias name. If not found, it will throw an error. If you want to use a character variable or expression, then you will need to surround it with parentheses.

* Two ways to select the workarea containing a table with an alias of MYTABLE
SELECT MyTable && Alias literal

A="MYTABLE"
SELECT (A) && Alias expression
SELECT (i+1) && Work area ID expression

Note

Traditional xBase systems limit the number of open work areas in a data session to 32767, but JAXBase limits it to available memory, with a maximum ID of 9007199254740992. JAXBase does not initialize a work area or data session until it is selected. When you select or otherwise move to a work area or data session, it is initialized and takes up memory space. The following code snippet explains the issue:

```
* Initializes 255 work areas needlessly takes
* up memory space for 255 work areas
FOR I=1 to 255
    SELECT (i)
ENDFOR
```

When a table or cursor is closed, most of the memory in use is released, but the initializing information remains. There is no way to release that memory in data session 1 without ending the application or executing a CLEAR ALL or CLOSE ALL command.

Data sessions greater than 1 can be CLOSED, releasing the memory it consumed. When you close a data session, all open databases and tables in that data session are closed. Issuing CLOSE ALL or CLOSE DATASESSION 1 closes all database connections, tables, and cursors, but the data session will remain open.

Forms that have a private data session will open the lowest available data session ID. Manually closing a form's data session will potentially generate numerous errors. Care should be taken to keep track of the data session IDs that you create with SELECT DATASESSION.

See Also

[ADATASESSIONS\(\) Function](#)
[ALIAS\(\) Function](#)
[AWORKAREAS\(\) Function](#)
[CLOSE Command](#)
[CLOSE DATASESSION Command](#)
[SELECT\(\) Function](#)

SET Commands

The SET command is used to set properties, features, or values in the JAXBase system. The following examples will explain each set command.

With all SET commands, no value provided when ON or OFF is expected is the equivalent of sending OFF.

Status

The documentation for the SET commands is incomplete and is slated for Version 0.4

SET ALTERNATE Command

```
SET ALTERNATE ON|OFF  
SET ALTERNATE TO <filename> [ADDITIVE][STAMP]
```

Remarks

Directs console or printer output created with the ?, ??, DISPLAY, or LIST commands to a text file.

Parameters

ON|OFF

If set on, the console or printer output will be saved to the indicated path and filename.

TO <filename>

If no filename is given, then the current filename is removed from memory and any more output will be lost until a new filename is provided.

ADDITIVE

Causes an existing file to be appended to rather than erased.

STAMP

Writes a timestamp to the file in the format Alternate printing set at HH:MM on YYYY-MM-DD.
Another timestamp will be written to a file if the STAMP and ADDITIVE are included with the filename.

Example

```
* Demonstrate SET ALTERNATE  
SET default to c:\temp  
SET ALTERNATE TO test  
SET ALTERNATE ON  
? "This is a test"  
SET ALTERNATE TO      && SET ALTERNATE OFF is assumed
```

Expected Output

The file C:\TEMP\TEST.TXT will be created with This is a test written to the file.

Notes

When setting a file name, if a path is not provided, it will add the default path. If there is no file extension, it will add .txt to the end. If an illegal path\filename is given or the file cannot be created, an error will be raised.

See Also

[FILETOSTR\(\) Function](#)

[STRTOFILE\(\) Function](#)

SET ASSERTS Command

Specifies if ASSERT commands are evaluated or ignored.

DRAFT

SET AUTOINCERROR Command

Specifies whether attempts to update values in a field using autoincrementing generate an error or fail silently and proceed. (Ignored until Autoincrement is supported).

DRAFT

SET AUTOSAVE Command

Determines whether JAXBase flushes data buffers to disk when you exit a READ or return to the Command window.

DRAFT

SET BELL Command

Turns the computer bell on or off and sets the bell attributes.

DRAFT

SET BLOCKSIZE Command

Specifies how JAXBase allocates disk space for the storage of memo fields.

DRAFT

SET BROWSEIME Command

Specifies if the Input Method Editor is opened when you navigate to a text box in a Browse window.

DRAFT

SET CARRY Command

Determines whether JAXBase carries data forward from the current record to a new record created with INSERT, APPEND, and BROWSE.

DRAFT

SET CENTURY Command

`SET CENTURY [ON|OFF] [TO nCentury]`

Remarks

Use SET CENTURY to specify how date variables and functions are displayed in JAXBase. Issuing SET CENTURY TO without any additional arguments restores the default century to the current century's digits (20) and displays all 4 digits of the year.

Parameters

ON

(Default) Specifies a four-digit year in a format that includes 10 characters (including date delimiters).

OFF

Specifies a two-digit year in a format that includes eight characters and assumes the current century for date calculations. This setting is not recommended.

TO nCentury

A number from 1 to 99 that specifies the current century. When a date has a two digit year, nCentury determines in which century the year occurs.

Note

SET CENTURY OFF always implies dates in the current century. However, the SET CENTURY TO syntax takes precedence over this setting.

The value of SET CENTURY TO is scoped to the application.

Setting Century to no value assumes the current century's digits (20).

See Also

[DATE\(\) Function](#)

[DATETIME\(\) Function](#)

[YEAR\(\) Function](#)

SET CLASSLIB Command

Opens a visual class library (.vcx) containing class definitions.

DRAFT

SET COLLATE Command

Specifies a collation sequence for character fields in subsequent indexing and sorting operations.

DRAFT

SET CONFIRM Command

Specifies whether the user can exit a text box by typing past the last character in the text box.

DRAFT

SET CONSOLE Command

```
SET CONSOLE [name] [SIZE <nCols>,<nRows> | FULL] [AT <nLeft>,<nTop>]
[ON | OFF | ACTIVE | INACTIVE]
```

Warning: This command departs from normal XBase behavior.

Typical XBase systems have a pallet where all forms will typically be displayed. If you resize the pallet or minimize it, the XBase program you are running will be affected. In JAXBase, each form is placed onto the desktop and can be moved anywhere. They are not tied to a visual environment. A console in JAXBase is a text-based area which can be resized and affected by various JAXBase functions that work with a console (ex: ?, ??, CLEAR, etc.).

Status

Full implementation slated for Version 2.0

Remarks

Enables or disables the console window and provides control over its size and position. At startup, the system creates a console window with the name DEFAULT. This console is set to 80 columns by 25 rows, positioned at the top left of the screen, and is not visible, and set to NODEBUG. In version 1.0, there is only one named console allowed.

Parameters

NAME

If not provided, assumes the DEFAULT console. In Version 1.0, all names are ignored and the DEFAULT console is affected.

SIZE <nCols>,<nRows> - Slated for Version 2.0

Resizes the console window as specified by the numeric column and row parameters. Initially opens at 80 columns by 25 rows. Minimum values allowed are 40 columns by 4 rows.

AT <nLeft>,<nTop>

Positions the console window on the desktop to the graphical position (in pixels) based on the numeric left and top parameters. 0,0 positions it at the upper left of the desktop.

ON

Makes the console visible.

OFF

Hides the console window.

DEBUG

Allows debug information to be displayed to the console if DEBUG or TALK settings are set ON.

NODEBUG

Prevents debug information from being displayed to the console no matter the DEBUG setting.

[Example](#)

[Note](#)

[See Also](#)

DRAFT

SET COVERAGE Command

Turns code coverage on or off, or specifies a text file to which code coverage information is directed.

DRAFT

SET CPCOMPILE Command

Specifies the code page for compiled programs.

DRAFT

SET CPDIALOG Command

Specifies whether the Code Page dialog box is displayed when a table is opened.

DRAFT

SET CURRENCY Command

Defines the currency symbol and specifies its position in the display of numeric, currency, float, and double expressions.

DRAFT

SET CURSOR Command

Determines whether the insertion point is displayed when JAXBase waits for input.

DRAFT

SET DATABASE Command

Specifies the current database.

DRAFT

SET DATASESSION Command

Activates the specified form's data session.

DRAFT

SET DATE Command

Specifies the format for the display of Date and DateTime expressions.

DRAFT

SET DEBUGOUT Command

Directs debugging output to a file.

DRAFT

SET DECIMALS Command

Specifies the number of decimal places displayed in numeric expressions.

DRAFT

SET DEFAULT Command

Specifies the default drive and directory.

DRAFT

SET DELETED Command

Specifies whether JAXBase processes records marked for deletion and whether they are available for use in other commands.

DRAFT

SET DEVELOPMENT Command

Causes JAXBase to compare the creation date and time of a program with those of its compiled object file when the program is run.

DRAFT

SET ESCAPE Command

Determines whether pressing the ESC key interrupts program and command execution.

DRAFT

SET EVENTLIST Command

Specifies events to track in the Debug Output window or in a file specified with SET EVENTTRACKING.

DRAFT

SET EVENTTRACKING Command

Turns event tracking on or off or specifies a text file to which event tracking information is directed.

DRAFT

SET EXACT Command

Specifies the rules JAXBase uses when comparing two strings of different lengths.

DRAFT

SET EXCLUSIVE Command

Specifies whether JAXBase opens table files for exclusive or shared use on a network.

DRAFT

SET FDOW Command

Specifies the first day of the week.

DRAFT

SET FIELDS Command

Specifies which fields in a table can be accessed.

DRAFT

SET FILTER Command

Specifies a condition that records in the current table must meet to be accessible.

DRAFT

SET FIXED Command

Specifies if the number of decimal places used in the display of numeric data is fixed.

DRAFT

SET FULLPATH Command

Specifies if DBF() and NDX() return the path in a file name.

DRAFT

SET FWEEK Command

Specifies the requirements for the first week of the year.

DRAFT

SET HEADINGS Command

Determines whether column headings are displayed for fields and whether file information is included when TYPE is issued to display the contents of a file.

DRAFT

SET HELP Command

Enables or disables Microsoft JAXBase online Help or specifies a Help file.

DRAFT

SET HOURS Command

Sets the system clock to a 12- or 24-hour time format.

DRAFT

SET INDEX Command

Opens one or more index files for use with the current table.

DRAFT

SET LOCK Command

Enables or disables automatic file locking in certain commands.

DRAFT

SET MACKEY Command

Specifies a key or key combination that displays the Macro Key Definition dialog box.

DRAFT

SET MEMOWIDTH Command

Specifies the displayed width of memo fields and character expressions.

DRAFT

SET MESSAGE Command

Defines a message for display in the main JAXBase window or in the graphical status bar, or specifies the location of messages for user-defined menu bars and menu commands.

DRAFT

SET MULTILOCKS Command

Determines whether you can lock multiple records using LOCK() or RLOCK().

DRAFT

SET NAMING Command

When a JAXBase command that contains a folder, database, or table name is executed, SET NAMING will alter the name in the following ways:

NATURAL – make no changes

LOWER – change to lower case

UPPER – change to upper case

PROPER – change to lower case except for first letter and any letter following a dash (-) or space.



SET NEAR Command

Determines where the record pointer is positioned after FIND or SEEK unsuccessfully searches for a record.

DRAFT

SET NOCPTRANS Command

Prevents translation to a different code page for selected fields in an open table.

DRAFT

SET NOTIFY Command

Enables or disables the display of certain system messages.

DRAFT

SET NULL Command

Determines how null values are supported by the ALTER TABLE, CREATE TABLE and INSERT - SQL commands.

DRAFT

SET NULLDISPLAY Command

Specifies the text displayed for null values.

DRAFT

SET ODOMETER Command

Specifies the reporting interval of the record counter for commands that process records.

DRAFT

SET ORDER Command

Designates a controlling index file or tag for a table.

DRAFT

SET PATH Command

Specifies a path for file searches.

DRAFT

SET POINT Command

Determines the decimal point character used in the display of numeric and currency expressions.

DRAFT

SET PROCEDURE Command

Opens a procedure file.

DRAFT

SET REFRESH Command

Determines whether and how often a Browse window is updated with changes made to records by other users on the network.

DRAFT

SET RELATION Command

Establishes a relationship between two open tables.

DRAFT

SET REPROCESS Command

Specifies how many times and for how long JAXBase attempts to lock a file or record after an unsuccessful locking attempt.

DRAFT

SET RESOURCE Command

Updates or specifies a resource file.

DRAFT

SET SAFETY Command

Determines whether JAXBase displays a dialog box before overwriting an existing file, or whether table or field rules, default values, and error messages are evaluated when changes are made in the Table Designer or with ALTER TABLE.

DRAFT

SET SECONDS Command

Specifies whether seconds are displayed in the time portion of a DateTime value.

DRAFT

SET SECURITY Command

Security at rest will become a concern starting with versions after JAXBase 1.0 and the SET SECURITY command will allow you to set levels of data encryption including NONE, 128, and 256 with yet-to-be-released options. More secure encryption options will, of course, slow down processing.

DRAFT

SET SEPARATOR Command

Specifies the character used to separate each group of three digits to the left of the decimal point in displaying numeric and currency expressions.

DRAFT

SET SKIP Command

Creates a one-to-many relationship among tables.

DRAFT

SET SKIP OF Command

Enables or disables a menu, menu bar, menu title, or menu item for user-defined menus or the Microsoft JAXBase system menu.

DRAFT

SET SPACE Command

Determines whether a space is displayed between fields or expressions when you use the ? or ?? command.

DRAFT

SET SQLCONNECTION Command

```
SET SQLCONNECTION TO sqlobject
```

Remarks

Specifies a connection to a backend SQL engine that will be used to create a default connection string. A SQL Object is created using the command CREATEOBJECT("SQLCONNECTION") which will create a JAXBase class that can be used to create a SQL connection.

Parameters

sqlObject

Variable name holding initialized SQL class.

Example

```
* CREATE THE SQL OBJECT
MyDB=CREATEOBJECT( "sql" )

* Just fill in the connection data
WITH MyDB
    .Driver="SQL Server"
    .Server="MyServer"
    .Port=1433      && Default SQL Server connection
    .USERID="sqladmin"
    .PASSWORD="BobIsAtHD-3167"
    .DATABASE="JAXCorp"
ENDWITH

* SET THE DEFAULT CONNECTION
SET SQLCONNECTION TO MyDB

* OPEN THE DATABASE AND BROWSE THE OFFICERS TABLE
OPEN DATABASE JAXCorp
SELECT * FROM OFFICERS
? RECCOUNT()
CLOSE DATABASE ALL
```

Notes

You can use JSON functions to easily save or load object properties.

A SQL Class object can be used by the OPEN DATABASE command even if the SQL object is already connected to the database. The only requirement is that the name used by the OPEN DATABASE command be a different name than the variable name holding the SQL object.

See Also

CREATE DATABASE
JSONTOOBJ()
OBJTOJSON()
GETJSONFIELD()
PUTJSONFIELD()

SET STEP Command

Opens the Trace window and suspends program execution for debugging.

DRAFT

SET STRICTDATE Command

Specifies if ambiguous Date and DateTime constants generate errors.

DRAFT

SET SYSMENU Command

Enables or disables the JAXBase system menu bar during program execution, and allows you to reconfigure it.

DRAFT

SET TABLEPROMPT Command

Enables or disables file open dialog from appearing when table cannot be located during execution of a data command such as SELECT - SQL Command.

DRAFT

SET TABLEVALIDATE Command

Specifies the level of table validation to perform.

DRAFT

SET TALK Command

```
SET TALK ON | OFF | CONSOLE | NOCONSOLE
```

DRAFT

SET TEXTMERGE Command

Enables or disables the evaluation of fields, variables, array elements, functions, or expressions that are surrounded by text-merge delimiters, and lets you specify text-merge output.

DRAFT

SET TEXTMERGE DELIMITERS Command

Specifies the text-merge delimiters.

DRAFT

SET TOPIC Command

Specifies the Help topic or topics to open when you invoke the JAXBase Help system.

DRAFT

SET TOPIC ID Command

Specifies the Help topic to display when you invoke the JAXBase Help system. The Help topic is based on the topic's context ID.

DRAFT

SET TRBETWEEN Command

Enables or disables tracing between breakpoints in the Trace Window.

DRAFT

SET TYPEAHEAD Command

Specifies the maximum number of characters that can be stored in the type-ahead buffer.

DRAFT

SKIP Command

SKIP [nExpr | TOP | BOTTOM]

Remarks

Advances the record pointer.

Parameters

nExpr

Numeric expression that indicates how many records to move down towards the bottom of the table if the value is positive, and how many records to move up towards the top of the table if the value is negative.

Example

```
* Demonstrate SKIP command
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
? RECNO()
SKIP 3
? RECNO()
```

Note

If SKIP is issued with no expression, the default is to move the record pointer one row down. If the SKIP command has a positive numeric expression, then the record pointer is moved down that many records. If negative, the record pointer moves up toward the beginning of the file that many records.

If the record pointer tries to go past the beginning of the table order, then BOF() will return True (.T.) and if it attempts to go past the end of the table order, EOF() will return True (.T.). If skip is issued with a positive value after the record pointer has reached the end of the table and EOF() is set to true, then an error is generated. The same is true for attempting to skip with a negative value when the pointer is already at the top of the file and BOF() is set to true.

See Also

GOTO Command
USE Command

SORT

```
SORT TO cTable ON FieldName1 [/A | /D] [/C]
[, FieldName2 [/D] [/C] ...]
[Scope] [FOR lExpression1]
[WHILE lExpression2]
[FIELDS FieldList | FIELDS LIKE Skeleton | FIELDS EXCEPT Skeleton]
```

Status

Slated for after Version 1.0

Remarks

Sorts the current table or cursor in the specified manner and stores the results to a table.

Parameters

TO cTable

Character literal specifying the name of the JAXBase table to save the results.

ON FieldName1

Specifies a field to use as a sort key.

/D [/C]

Modifies the sort. /D indicates that the field is to be sorted in descending order. /C indicates that the field is to be sorted as case insensitive. If the field is not a character field, /C is ignored.

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression limiting the records used in AVERAGE while the expression is true.

FIELDS FieldList

Specifies the fields to populate from the array.

FIELDS LIKE

Specifies a field skeleton where matching fields are populated from the array.

FIELDS EXCEPT

Specifies fields to skip when populating from an array

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
SORT TO P203 ON Party /D, Precinct FOR Precinct=203
USE P203
BROWSE
```

Note

You cannot sort on a MEMO, GENERAL, or BLOB field.

FieldName cannot be an expression. It must be a field name.

The SORT command is quite slow compared to using indexes.

See Also

BROWSE
Using Indexes

STORE

```
STORE eExpression1 [,eExpression2...] TO cVar1 [, cvar2...]
```

Remarks

Assigns a value to a variable or property, or a list of values to a list of properties.

Parameters

eExpression1 [, eExpression2...]

One or more expressions that evaluate to JAXBase data types, objects, or .NULL.

TO cVar1 [, cVar2...]

One or more character literal names of variables to which the expressions are assigned.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
STORE 0,1,2 TO A,B,C
? A,B,C
```

Note

There must be the same number of expressions as variable names, or an error will be generated.

If you assign a value to an array instead of an array element, then that value is assigned to all elements of the array.

See Also

Arrays, Variables, and Objects

SUM

```
SUM [ExprList] [Scope] [FOR lExpr1] [WHILE lExpr2]
    [TO VarList | TO ARRAY ArrayName][ADDITIVE]
    [IN nWorkArea | cTableAlias]
```

Remarks

Takes the averages of the ExprList and puts them into VarList or an array.

Parameters

ExprList

Comma delimited list of expression to use when creating the averages to place into VarList.

Scope

Specifies the range of records to use. Only records that fall within the scope are used. Scope clauses are: ALL, NEXT nRecords, RECORD nRecordNo, REST, and TOP nRecords. The default scope is ALL.

FOR lExpr1

Logical expression used to filter what records are used.

WHILE lExpr2

Logical expression limiting the records used in AVERAGE while the expression is true.

TO VarList

Character literal list of variable names that will receive the averages.

TO ArrayName [ADDITIVE]

Character literal of array name to create or append to if ADDITIVE is provided.

IN nWorkArea | cAlias

Numeric work area literal or character literal for alias name of table to use.

Example

```
* Demonstrate AVERAGE
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
USE (_JAXFolder+"data\polldata")

* If there are no matching records, all non-literal numeric
* expressions will return 0.00 as their result.
SUM age TO ARRAY aVoteAge for Precinct=203 AND Party="I"
SUM age to ARRAY aVoteAge ADDITIVE for Precinct=203 AND Party="D"
SUM age to ARRAY aVoteAge ADDITIVE for Precinct=203 AND Party="R"

DISPLAY MEMO LIKE aVoteAge
```

Note

When saving to an array, if the array does not exist then it is created. If it does exist, it is overwritten with the corresponding number of elements. If ADDITIVE keyword is included, then the array is appended with a new row. If the number of expressions does not match the existing array column count, an error is generated. If there are no results, the array is updated numeric literals and any non-literal expressions are set to 0.00.

ADDITIVE should not be in the first of a series of SUM statements.

If saving to a variable list, values are returned even if there are no matching records, in which case 0.00 is filled in for any non-literal numeric expression.

See Also

AVERAGE Command
CALCULATE Command
COUNT Command
TOTAL Command
Arrays, Variables, and Objects

SUSPEND

SUSPEND

Status

Slated for after Version 1.0

Remarks

If in the IDE, causes the program to suspend and brings up the Debug Window.

Parameters

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Note

If the program is not running in the IDE, then the SUSPEND command is ignored.

See Also

ASSERT Command
CANCEL Command
RESUME Command
QUIT Command
Debug Window

TEXT / ENDTEXT

```
TEXT [to cVar [ADDITIVE][TEXTMERGE][NOSHOW][PRETEXT eExpression]]
      Merge Text
ENDTEXT
```

Status

Slated for Version 1.0

Remarks

Parameters

Merge Text

Text lines that are used to create the resulting text merge.

TO cVar

Character literal of variable name to store the resulting merged text.

ADDITIVE

If ADDITIVE is included, the resulting merged text is added to the contents of cVar, otherwise it is overwritten.

TEXTMERGE

The TEXTMERGE parameter causes the evaluation of merge fields in the merge text.

NOSHOW

By default, when the TEXT command is executed, the results are sent to the current output device (typically the console). The NOSHOW parameter prevents that from happening.

PRETEXT eExpression

If eExpression is a character expression, insert it before each line of the results.

If eExpression is a numerical expression, use the following table to understand its purpose.

Value (Additive)	Description
1	Eliminate spaces before each line
2	Eliminate tabs before each line
4	Eliminate carriage returns
8	Eliminate line feeds

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")

TEXT TO cText TEXTMERGE NOSHOW
Dear <<Name>,

We have you listed as <<IsParty()>> and would like to remind you that
you live in voting precinct <<Precinct>>. Your voting location is located
<<VotingLoc>>.

Thank you,

Jack B Nimble
Office of Voter Registration
ENDTEXT

? cText
USE
RETURN

PROCEDURE IsParty
RETURN ICASE(Party="R", "Republican", Party="D", "Democrat", "Independent")
```

Note

The text merge delimiters are set using the SET TEXTMERGE DELIMITERS command.

When a textmerge field is evaluated, the results are trimmed of leading and trailing spaces.

If a textmerge field cannot be evaluated, an error is generated.

You have two ways of sending the results of the text command to a device.

You can capture the results in a variable using the TO parameter and sending the value to a file, printer, or console.

You can direct console output to an alternate file or printer.

While textmerge can be used to create letters, there are more modern methods discussed at Merging Text.

See Also

[SET MEMOWIDTH Command](#)
[SET TEXTMERGE Command](#)
[SET TEXTMERGE DELIMITERS Command](#)

THROW

```
THROW nError [, cMessage]
```

Status

Slated for Version 1.0

Remarks

Allows you to create a custom error message.

Parameters

nError

Numeric expression between 0 and 999 or 7000 and 7999.

cMessage

Character expression containing the user defined message.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

TRY
    THROW 0, "TEST ERROR"
CATCH TO loErr
    ? loErr.ErrorNo,loErr.Message
ENDTRY
```

Note

If the nError is less than 1000 then 7000 is added to it and the result is a user defined error between 7000 and 7999. If nError is not less than 1000 or between 7000 and 7999, a "Value or index is out of range" error is generated.

The original values in nError and cMessage are written out to the log file, if it's active.

See Also

ON ERROR Command
 TRY/CATCH/FINALLY/ENDTRY Command
 ERROR Event

TRY/CATCH/FINALLY/ENDTRY

```

TRY
    Command block
CATCH [TO cVar]
    Command block
FINALLY
    Command block
ENDTRY

```

Remarks

Creates a structured error handling system to handle errors for a block of code.

Parameters

TRY

Beginning of the TRY/CATCH/FINALLY/ENDTRY structure.

CATCH

Beginning of the CATCH block which is executed when an error occurs in the command block following the TRY statement.

TO cVAR

Creates an object variable named after the character literal name that contains the following properties.

Property	Data Type	Description
ErrorNo	Numeric	Error number
LineNo	Numeric	Line where error occurred
Message	Character	Error Message
Procedure	Character	Procedure where error occurred (may be catching an error from a UDF)
StackLevel	Numeric	Program level where error occurred. Especially used when performing recursion.

FINALLY

Start of finally block which is usually cleanup code to close resources opened in the TRY block.

ENDTRY

End of TRY/CATCH/FINALLY/ENDTRY structure.

Example

```

* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

TRY
    FOR i = 3 to -1
        DO TEST1 WITH i
    ENDFOR
CATCH TO loErr
    ? "Error",TRANSFORM(loErr.ErrorNo) "@ line",TRANSFORM(loErr.LineNo),"in",loErr.Procedure
ENDCATCH
RETURN

* Will thrown an error when passed -1
PROCEDURE TEST1
PARAMETERS tnIdx
? SUBSTR("ABCDEFG",tnIdx,1)
RETURN

```

Note

The CATCH block only executes if an error occurs which is passed to the CATCH command.

The FINALLY code block only executes if the TRY or CATCH block successfully executes and a QUIT or CANCEL command is not executed in either command block.

If an error occurs in the CATCH block, an error is generated that needs to be caught TRY/CATCH/FINALLY/ENDTRY in the CATCH block, an ON ERROR command, or an Error event. Failing that, the program will terminate.

A program, UDF, or method/event that was called from the TRY/CATCH/FINALLY/ENDTRY structure that generates an error and does not have its own error handler, sends the error to the CATCH block.

See Also

ON ERROR Command
Error Event

UNLOCK

UNLOCK [RECORD nRecord] [IN nWorkArea | cAlias] [ALL]

Status

Slated for Version 1.0

Remarks

Releases a locked record, multiple records, a file lock on a table, or all locks on all open tables.

Parameters

RECORD nRecord

Numeric expression specifying record to unlock.

IN nWorkArea | cAlias

Numeric work area literal or character literal of alias name of table to unlock.

ALL

Releases all locks on all tables.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata")
? LOCK(0)
? RLOCK(5)
? ISFLOCKED()
UNOCK ALL
? ISFLOCKED()
```

Note

Specifying record 0 releases a header lock, otherwise releases a record lock for a specific record.

Record locks can only be released by the user that creates the locks.

See Also

[FLOCK\(\) Function](#)
[LOCK\(\) Function](#)
[RLOCK\(\) Function](#)

UPDATE

```
UPDATE cSQLTable
SET Column_Name1 = eExpression1 [, Column_Name2 = eExpression2 ...]
[FROM [FORCE] Table_List_Item [[, ...] | [JOIN [ Table_List_Item]]]
WHERE FilterCondition
```

Status

Slated for Version 2.0

DRAFT

UPDATE FROM

UPDATE FROM nWorkArea | cAlias TO SQLTable

Remarks

Updates a SQL Table using records from the indicated alias or work area.

Parameters

FROM nWorkArea | cAlias

Numeric work area literal or character literal for alias name of an open table or cursor.

TO SQLTable

Character literal specifying the name of the SQL table to update.

Example

```
* Assumes a SQL Database has been opened and
* that it contains a table that has the same
* field names and compatible types as the
* TEST_CCSV table created here.
CREATE TABLE TEST_CCSV (PKEY C(10), Name C(30), BirthDate D, Sex C(1), Height N(4,1))
APPEND FROM (_JAXFolder+"samples\data\PEOPLE.CSV") TYPE CSV

* Insert code to create valid Primary Key values for PKEY
* in each record. If the primary key matches an existing
* primary key, that record's fields will be overwritten
* with the fields from the table. If the primary key
* does not match, then the record is appended to the table.

* Update the SQL Table
UPDATE FROM 0 TO MySQLTable
```

Note

The FROM parameter is required.

The FROM parameter of 0 means the current work area.

TO SQLTable assumes the current database. To specify the database, use the Database!Table naming format.

See Also

[APPEND FROM Command](#)

[OPEN DATABASE Command](#)

[CREATE TABLE/CURSOR Command](#)

[Moving Data To and From a SQL Database](#)

USE

OPEN A DBF

```
USE cTable [IN nWorkArea] [ALIAS cAlias] [AGAIN]
  [[INDEX cIndex] [ASCENDING | DESCENDING][[, cIndex] [ASCENDING | DESCENDING]...]]
  [EXCLUSIVE] [SHARED] [NOUPDATE]
```

RETRIEVE DATA FROM A SQL TABLE (Slated for Version 1.0)

```
USE cDbName!Table [IN nWorkArea] [ALIAS cAlias] [WHERE lExpression] [NOUPDATE]
```

Remarks

The first command opens a local DBF table while the second command retrieves records from a SQL database table.

Parameters

cTable

Character literal of table to open.

IN nWorkArea

Numeric work area literal of table to use.

ALIAS cAlias

Character literal of alias to use for table or cursor.

AGAIN

A DBF cannot be opened again once unless you use the AGAIN keyword on the subsequent USE commands.

INDEX cIndex

Character literal of the index to use.

ASCENDING | DESCENDING

Default is ASCENDING. DESCENDING reverses the order of the index.

EXCLUSIVE

Open the table for exclusive access. Required if SET EXCLUSIVE is OFF and you do not wish to share the table.

SHARED

Open the table for shared access. Required if SET EXCLUSIVE is ON and you wish to share the table.

NOUPDATE

Opens the table for READONLY access.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\polldata") ALIAS Polls
? ALIAS()
? DBF()
? RECNO()
? RECCOUNT()
```

Note

If you omit the WHERE command when retrieving records from a SQL table, you will load all records into a cursor, potentially causing a long pause in execution.

ASCENDING and DESCENDING can be abbreviated as ASC and DES.

NOUPDATE will only set the table to read only for that USE statement. If you have the table open under other multiple aliases, they are not affected.

You can open any number of indexes, and you can also open additional indexes using the SET INDEX command.

Indexes are assumed to be in the same folder as the table. If you provide a path to the indexes, it will use that path. If you don't supply a path and the indexes are not found, JAXBase will search the path list until it finds a match.

See Also

- AVERAGE Command
- CALCULATE Command
- CLOSE Command
- CLEAR Command
- DELETE Command
- OPEN DATABASE Command
- RECALL Command
- REPLACE Command
- SET INDEX Command
- SUM Command

WAIT

WAIT [cExpression] [TO cVar] [TIMEOUT nSeconds] [CLEAR] [NOWAIT]

Status

Slated for Version 1

Remarks

Displays a message window in the upper right corner of the monitor.

Parameters

cExpression

Character expression of message to display.

TO cVar

Character literal of variable name to return the integer value of the key press. See INKEY() for values.

TIMEOUT nSeconds

nSeconds is a numeric expression indicating how many seconds to wait before clearing the message.

CLEAR

Removes the window from the display area.

NOWAIT

Display the message and continue execution. The message is displayed until WAIT CLEAR is or CLEAR command is issued.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
WAIT WINDOW "Press a key" to lnKey
? lnKey
```

Note

Omitting the TIMEOUT clause is the same as TIMEOUT 0, which keeps the window displayed until a key is pressed or a CLEAR command is issued..

If TO cVAR is included, the NOWAIT clause is ignored, if present.

See Also

CLEAR Commands

WITH/ENDWITH

```
WITH ovar
    Command block
ENDWITH
```

Status

Slated for Version 1

Remarks

Specifies what object to use with properties in a block of code.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

WITH _MOUSE
    ? .XPos
    ? .YPos
ENDWITH
```

Note

WITH allows you a short-cut to working with an object by assuming that any .cName is a property that belongs to that object. If the .cName literal expression is not a property of the object in the WITH command, an error is generated.

See Also

[ADD CLASS Command](#)
[CREATE CLASS Command](#)
[CREATEOBJECT\(\) Command](#)

ZAP

```
ZAP [ IN nWorkArea | cAlias ]
```

Remarks

Removes all records from a table and causes all open indexes to be reindexed.

Note

ZAP is the same thing as DELETE ALL followed by PACK, but ZAP is faster.

A table must be used in EXCLUSIVE mode to ZAP, otherwise an error is generated.

You cannot ZAP a cursor that is in READONLY mode.

See Also

[DELETE Command](#)

[PACK Command](#)

DRAFT

Function Reference

DRAFT

ABS()

ABS(nExpr)

Remarks

Returns the absolute value of a numeric expression.

Parameters

nExpr

Numeric expression that will be converted to its absolute value.

Example

? ABS(-1) && Returns 1

Return Value

Numeric

See Also

[INT\(\) Function](#)

[ROUND\(\) Function](#)

[SIGN\(\) Function](#)

[ISODD\(\) Function](#)

ACLASS()

ACLASS(ArrayName, ClassObject)

Remarks

Returns an array containing the hierarchy of the class names.

Parameters

ArrayName

The name of the array to place the results. If the array name does not exist, it is created. If the name exists, it is overwritten to the correct size. The array will be returned as a one-dimensional array.

ClassObject

The Object you wish to interrogate.

Example

```
* Create two custom classes and get the
* names of the class hierarchy
CLEAR
lnCount=ACCLASS(laNames, "MyTextBox")
DISPLAY MEMORY LIKE mytextbox
                                         && Displays the arrays elements

DEFINE CLASS MYTXBOX AS TEXTBOX
ENDDEFINE

DEFINE CLASS MYTEXTBOX AS MYTXBOX
ENDDEFINE
```

Return Value

Numeric

Note

Returns 0 if the array cannot be created, otherwise the number of elements in the array.

See Also

[ADD CLASS Command](#)

[AMEMBERS\(\) Function](#)

[CREATE CLASS Command](#)

[CREATE OBJECT Command](#)

[DEFINE CLASS Command](#)

ACOPY()

```
ACOPY(SourceArray, TargetArray [,nFirstSourceElement [,nCount [,nFirstTargetElement]]])
```

Remarks

Copies elements from one array to another.

Parameters

SourceArray

The array to copy from.

TargetArray

The array to copy to.

nFirstSourceElement

Which element to start with in the source array.

nCount

Number of elements to copy.

nFirstTargetElement

The starting element in the target array where the copy will be placed.

Example

```
* Test the ACOPY() function
nPCount=APRINTERS(aPrnt)
DIMENSION aNewList[20]
```

```
IF PCount>0
  IF pCount>20
    =ACOPY(aPrnt,aNewList,1,20)
  ELSE
    =ACOPY(aPrnt,aNewList)
  ENDIF
ENDIF
```

```
DISPLAY MEMO LIKE aNewList
```

Return Value

Numeric

Note

When copying from an array to an existing array, the target array's size is not altered, and you need to make sure that there is enough room for the copy.

The function copies from one array to the other as if they are one-dimensional, meaning it looks at the number of elements in the array and not the number of rows and columns.

If the array does not exist, it will be created. If the destination is not an array or large enough to accept the number of elements, an error will be generated.

See Also

[ADEL\(\) Function](#)

[AELEMENT\(\) Function](#)

[AINS\(\) Function](#)

[ASCAN\(\) Function](#)

[ASORT\(\) Function](#)

[How JAXBase Deals with Arrays](#)

ACOS()

ACOS(nExpression)

Remarks

Returns the arc cosine of the numeric expression.

Parameters

nExpression

The numeric expression to use whose arc cosign is returned. The value of nExpression can start at -1 and go as high as +1. The value returned ranges from zero to PI (3.14159265)

Example

```
? ACOS(.5)           && Returns 1.05  
? RTOD(ACOS(.5))    && Returns 60
```

Return Value

Numeric

Note

If the nExpression is out of range, an error will be generated.

See Also

[ASIN\(\) Function](#)

[ATAN\(\) Function](#)

[ATN2\(\) Function](#)

[ACOS\(\) Function](#)

[DTOR\(\) Function](#)

[RTOD\(\) Function](#)

[SIN\(\) Function](#)

[TAN\(\) Function](#)

[SET DECIMALS Command](#)

ADATABASES()

ADATABASES (ArrayName)

Remarks

Returns a two dimensional array containing the names of open databases in column one, the Server\Instance name in column two, and the database version in column three.

Parameters

ArrayName

Name of the array to place the results.

Example

```
* Get a list of all open databases  
nCount=ADATABASES(aDBList)  
DISPLAY MEMORY LIKE aDBList
```

Return Value

Numeric

Note

If the array does not exist, it is created. If the name exists, it is overwritten and the corresponding number of rows added. If no databases are open, the array is not created and if the name exists, it is not altered.

See Also

CREATE DATABASE Command

DISPLAY DATABASE Command

LIST DATABASE Command

OPEN DATABASE Command

SQL Class

ADBOBJECTS()

ADBOBJECTS(ArrayName, cSetting)

Remarks

Places the names of tables or views in the current database to an array.

Parameters

ArrayName

Specifies the name of the array to place the results. If the array name does not exist, it will be created. If it does exist, it will be overwritten, and an array of the correct size will be created.

A one dimensional array is created for VIEW and TABLE settings. A two dimensional array is created for INDEX containing , for each index found, the table, the name of the index, the type of index, and the expression used to create the index.

cSetting

Character expression "index","table" or "view" indicating which to return.

Example

- * Interrogate the JAXCorp database that you installed
- * onto your MySQL or SQL Server database server using
- * the CreateDSNFile program in the TOOLS folder.

```
CLEAR ALL
CLOSE ALL
CLEAR
```

```
SET NAMING TO LOWER
cConnect=DECRYPT(FILETOSTR(_JAXFolder+"\tools\dsn\JAXCorp.dsnx"))
OPEN DATABASE JAXBase USING cConnect
nCount=ADBOBJECTS(gaTables,"table")
DISPLAY MEMORY LIKE gaTables      && List of all tables
```

Return Value

Numeric

Note

ADBOBJECTS() will return information for the currently open database. If there is no database open, then an error will be generated.

This function allows you to directly interrogate the database and retrieve information regarding the data definition in the dialect of the database engine. Notably when using the INDEX setting, you will receive the expression used to create the index using the syntax of that database engine. MySQL and SQL Server expressions may differ, so care must be taken when writing code to move a table from one server to the other.

A database opened by creating a SQL Class is addressable by the SET DATABASE command using the name property of the SQL Connection. As an example:

```
* Open a database using a SQL Class
oSQL1=CREATEOBJECT( "SQL" )
WITH oSQL1
    .Name="SQLDB1"
    .LoadDSN(DECRYPT(FILETOSTR(_JAXFolder+"\tools\dsn\JAXCorp.dsnx")))
    .Connect()
ENDWITH

IF oSQL1.Connected()=false
    ? "Error opening JAXCorp"
ENDIF

* Open the JAXSample database using a dsnx file
cConnect=DECRYPT(FILETOSTR(_JAXFolder+"\tools\dsn\JAXSample.dsnx"))
OPEN DATABASE JAX1 USING cConnect

? DBC()      && Displays JAXSAMPLE

SELECT DATABASE (oSQL1.Name)
? DBC()      && Displays JAXCORP
```

See Also

[ADATABASES\(\) Function](#)
[CREATE Command](#)
[CREATE DATABASE Command](#)
[CREATE TABLE - SQL Command](#)
[CREATOBJECT\(\) Function](#)
[DBC\(\) Function](#)
[DISPLAY DATABASE Command](#)
[INDBC\(\) Function](#)
[LIST DATABASE Command](#)
[OPEN DATABASE Command](#)
[SET DATABASE Command](#)
[SQL Class](#)

ADDBS()

ADDBS(cPath)

Remarks

Adds a backslash to a path if it doesn't already have one.

Parameters

cPath

A string expression that evaluates to a path.

Example

```
? ADDBS( "C:\WINDOWS" )      && Returns C:\WINDOWS\
```

Return Value

Character

Note

ADDBS() does no error checking, so if you send an invalid path, you'll get the invalid path back.

ADDBS() is different from JUSTFULLPATH() which returns the path with a backslash, because if you send "c:\temp" to ADDBS() you'll get "c:\temp\" back where JUSTFULLPATH() would return "c:\".

See Also

[DEFAULTEXT\(\) Function](#)

[FILE\(\) Function](#)

[FORCEEXT\(\) Function](#)

[FORCEPATH\(\) Function](#)

[JUSTDRIVE\(\) Function](#)

[JUSTTEXT\(\) Function](#)

[JUSTFULLPATH\(\) Function](#)

[JUSTFNAME\(\) Function](#)

[JUSTPATH\(\) Function](#)

[JUSTSTEM\(\) Function](#)

ADDPROPERTY()

ADDPROPERTY(oObjectName, cPropertyName, [eExpression])

Remarks

Adds a property to an existing object at run time.

Parameters

oObjectName

Object variable containing the class that you want to which you want to add a property.

cPropertyName

The property name you wish to add as a string expression.

eExpression

The optional value you wish to assign to the new property.

Example

```
* Demonstrate ADDPROPERTY()
USE (_JAXFolder+"data\phonebook")
SCATTER NAME oEntry
ADDPROPERTY(oEntry, "NewProperty", "HI!")
? oEntry.NewProperty           && Returns HI!
```

Return Value

Logical True (.T.) means the property was successfully added.

Note

If you are adding an array to an already existing array property, the old property is overwritten as an array with the new dimensions and values. If you attempt to overwrite an array property with a non-array value, all elements in that array will be set to that value.

All new properties are added as visible, read/write, and can change data types.

You cannot use ADDPROPERTY() to overwrite a read-only property (like BASECLASS) and you cannot use it to change the data type of a native property (like trying to change NAME from character to an integer value).

ADDPROPERTY() will return a false (.F.) value if you attempt to update a hidden or protected property.

See Also

FOREACH

SCATTER Command

REMOVEPROPERTY() Function

AddProperty Method

ADEL()

ADEL(ArrayName, nElementNumber [,1|2])

Remarks

Deletes an element from a one-dimensional array or a row or column from a two-dimensional array but does not change the size of the array.

Parameters

ArrayName

Specifies the array to alter.

nElementNumber

Specifies which element, row, or column to remove.

1|2

With a two-dimensional array, the value of 1 will delete the row while 2 will delete the specified column.

Example

```
* Demonstrate ADEL()
DIMENSION aMyArray[5,5]
FOR i=1 to 5
    FOR j=1 to 5
        aMyArray(i,j)=(i-1)*5+j
    ENDFOR
ENDFOR
* Displays 25 element values ranging from 1 to 25
DISPLAY MEMORY LIKE aMyArray

ADEL(aMyArray,1,1)
ADEL(aMyArray,5,2)

* Displays 25 element values but row 5 is filled with
* a false value (.F.) as is column 5
* Values 1-5 plus 10, 15, 20, and 25 are now missing
DISPLAY MEMORY LIKE aMyArray
```

Return Value

Numeric

Note

If successful, returns a numeric value of 1, otherwise returns 0.

ADEL does not change the size of the array. The trailing elements are moved toward the front. In a one-dimensional array, the trailing elements move up and last element is set to a value of false (.F.).

In a two-dimensional array, deleting a column causes the trailing columns move left and the last column is filled with a value of false (.F.) while deleting a row causes the trailing rows to move up and the last row is filled with a value of false (.F.).

See Also

[ACOPY\(\) Function](#)
[ADIR\(\) Function](#)
[AELEMENT\(\) Function](#)
[AFIELDS\(\) Function](#)
[AINS\(\) Function](#)
[ALEN\(\) Function](#)
[ASCAN\(\) Function](#)
[ASORT\(\) Function](#)
[ASUBSCRIPT\(\) Function](#)
[DIMENSION Command](#)

ADIR()

```
ADIR(ArrayName [,cFileSkeleton, [,cAttribute, [,nFlag]]])
```

Remarks

Puts information about files in the default path, or the specified location, and returns the number of rows returned in the array.

Parameters

ArrayName

Name of array to create.

cFileSkeleton

A character expression of a valid path and file name. The file name can consist of wildcards.

cAttribute

String expression containing any combination of the letters D, H, and S where D returns subdirectory names, H returns hidden file names, and S returns System file names. By default, directory names, hidden and system files are not returned.

nFlag

Numeric expression indicating case of filenames returned.

nFlag	Description
0	(default) Returns the filename in original case
1	Returns the filename in uppercase
2	Returns the filename in lowercase
10	Returns the full path and filename in original case
11	Returns the full path and filename in upper case
12	Returns the full path and filename in lower case

Example

```
* Demonstrate the ADIR() function
CLEAR ALL
CLOSE ALL
CLEAR

ADIR(aMyDir,_JAXFolder+"data\*.dbf")

* Displays file name, size, date last modified, time last modified, attributes
DISPLAY MEMO LIKE aMyDir
```

Returns

Numeric

Note

If the array name does not exist, it is created. If the name exists, it is overwritten, and the corresponding rows and columns are created.

If the path is invalid or unavailable, an error is generated.

If there are no matching files or folder entries, then the array is not created.

See Also

[ADEL\(\) Function](#)

[AELEMENT\(\) Function](#)

[AFIELDS\(\) Function](#)

[AINS\(\) Function](#)

[ALEN\(\) Function](#)

[ANETRESOURCES\(\) Function](#)

[ASCAN\(\) Function](#)

[ASORT\(\) Function](#)

[ASUBSCRIPT\(\) Function](#)

[DIMENSION Command](#)

[DIR or DIRECTORY Command](#)

AELEMENT()

AELEMENT(ArrayName, nRowSubscript [, nColumnSubscript])

Remarks

Returns the element number of an array from the provided subscript information.

Parameters

Arrayname

Name of the array whose element you want returned.

nRowSubscript

A numeric expression indicating the row of a two-dimensional array or the element of a one-dimensional array.

nColumnSubscript

A numeric expression indicating the column of a two-dimensional array.

Example

```
* Demonstrate the AELEMENT() function
DIMENSION aTest1[3,7]

? AELEMENT(aTest1,2,3)      && Returns 10
? AELEMENT(aTest1,3,7)      && Returns 21
```

Returns

Numeric

Note

AELEMENT() with invalid subscript references generates an error.

AELEMENT() with a missing column on a two-dimensional array will generate an error.

AELEMENT() on a one-dimensional array will return the same number as the nRowSubscript.

Attempting to use AELEMENT against a non-array variable or property will generate an error.

All arrays are stored as one-dimensional arrays in JAXBase. A two-dimensional array can be addressed as a two-dimensional array or as a one-dimensional array. Elements in a two-dimensional array start at 1,1 and move right across the row, then down a row and at column 1. As an example, a two-dimensional array named aTest with 2 columns and 2 rows is arranged in memory as:

Array aTest

Row	Col	Element
1	1	1
1	2	2
2	1	3
2	2	4

Thus, the above two-dimensional array can be represented as aTest[3] or aTest[2,1] to get the same element value.

See Also

[ADEL\(\) Function](#)

[ADIR\(\) Function](#)

[AFIELDS\(\) Function](#)

[AINS\(\) Function](#)

[ALEN\(\) Function](#)

[ASCAN\(\) Function](#)

[ASORT\(\) Function](#)

[ASUBSCRIPT\(\) Function](#)

[DIMENSION Command](#)

[DISPLAY MEMORY Command](#)

[How JAXBase Deals with Arrays](#)

AERROR()

AERROR(ArrayName)

Remarks

Puts one or more rows of error information from the error stack into the specified ArrayName and returns the number of rows in the array.

Parameters

ArrayName

Name of the array to create with information from the error stack.

Example

```
* Test the AERROR() function
TRY
  DIMENSION aTest[3]
  aTest[4]=7
CATCH
  AERROR(aErrInfo)
  DISPLAY MEMORY LINE aErrInfo      && Displays the captured error information
ENDTRY
```

Returns

Numeric

Note

When a JAXBase error occurs, the array contains one row.

Element	Description
1	Numeric. The error number returned, which is identical to ERROR().
2	Character. The text of the error message, identical to MESSAGE().
3	Empty character string unless there is an error parameter which is then returned.
4	Numeric 0, except as appropriate returns the workarea number.
5	Numeric 1, except as appropriate returns the data session number.
6	Character value of the program where the error occurred.
7	Numeric value of the line where the error occurred.

When an error occurs in an ODBC call, the array contains one or more rows for each ODBC error.

Element	Description
1	Numeric. The error number returned, which is identical to ERROR().
2	Character. The text of the error message, identical to MESSAGE().
3	Character. The message related to the ODBC error.
4	Character. The current ODBC SQL state.
5	Numeric. The error number from the ODBC source.
6	Character. The name of the SQL handle.
7	Character. The name of the database.

The following table represents a function that is slated for Version 2

When a communication error occurs, the array contains one or more rows for each error returned.

Element	Description
1	Numeric. The error number returned, which is identical to ERROR().
2	Character. The text of the error message, identical to MESSAGE().
3	Character. The error text returned from the called program.
4	Character. The name of the program called.
5	Character. The empty string or additional error information from the called program.
6	Character. The address of the called program (such as IP address).
7	Character. The empty string or other connection information related to the address.

See Also

[ERROR Command](#)

[ERROR\(\) Function](#)

[MESSAGE\(\) Function](#)

[ON ERROR Command](#)

[OPEN DATABASE Command](#)

[JAXBase Error Messages](#)

JAX Classes:

BARCODE, FTP, HTTP, IPC, PIP, POP3, PRINTER, SMPT, SMS, SQL, TCP, and UDP

AEVENTS()

Remarks

Returns an array containing existing event bindings between objects.

Status

Slated for Version 2

DRAFT

AFIELDS()

AFIELDS(ArrayName [,nWorkArea | cAlias])

Remarks

Creates an array containing the structure of the table and returns the number of rows created.

Parameters

ArrayName

Name of the array to create and store the structure.

nWorkArea

A numeric expression for the workarea.

cAlias

A character expression for an open table or cursor alias.

Example

```
* Demonstrate the AFIELDS() function
CLOSE ALL
CLEAR ALL
USE (_JAXFolder+"data\phonebook")
AFIELDS(aStructure)

DISPLAY MEMORY LIKE aStructure          && Returns field structure
```

Returns

Numeric

Note

The array will contain the following information for each JAXBase field.

Column	Data Type	Description
1	Character	Field Name
2	Character	Field Type – See JAXBase Field Types
3	Numeric	Field Width
4	Numeric	Precision (number of decimal places)
5	Logical	.F.
6	Logical	.F.
7	Character	Empty String
8	Character	Empty String
9	Character	Empty String
10	Character	Empty String
11	Character	Empty String
12	Character	Empty String
13	Character	Empty String

14	Character	Empty String
15	Character	Empty String
16	Character	Empty String
17	Numeric	NextValue for autoincrementing (Slated for Version 2)
18	Numeric	Step for autoincrementing (Slated for Version 2)

You can retrieve a SQL table's structure in JAXBase format or the native SQL format by using the AFIELDS() method in the SQL class.

See Also

[ALEN\(\) Function](#)

[ALTER TABLE - SQL Command](#)

[COPY STRUCTURE EXTENDED Command](#)

[CREATE Command](#)

[DIMENSION Command](#)

[SQL Class](#)

[JAXBase Field Types](#)

AFONT()

AFONT(ArrayName [,cFontName])

Remarks

Places available font information into an array.

Parameters

ArrayName

Name of array to place the font information into.

cFontName

Specifies the name of the font for which information is placed into the array.

Example

```
* Demonstrate the AFONT() function
```

```
CLOSE ALL
```

```
CLEAR ALL
```

```
? AFONT(afInfo)
```

```
LIST MEMO LIKE afInfo
```

Note

Due to differences in Windows and Linux, the AFONT() function simply retrieves font information with name as the only filter.

If the cFontName parameter is not passed, all available font information is loaded to the array.

See Also

[TXTWDITH\(\) Function](#)

AINS()

AINS(ArrayName, nElementNumber [, 2])

Remarks

Inserts an element into a one-dimensional array, or a row or column into a two-dimensional array.

Parameters

ArrayName

Name of element that will be modified.

nElementNumber

Specifies where the new element, row, or column is inserted into the array.

1|2

The value 1 specifies that a row is to be inserted while 2 specifies that a column will be inserted into a two-dimensional array.

Example

```
DIMENSION aMyArray[5,4]
FOR i=1 to 4
    FOR j=1 to 4
        aMyArray(i,j)=(i-1)*4+j
    ENDFOR
ENDFOR
AINS(aMyArray,2)
AINS(aMyArray,2,2)
```

The following is displayed:

AMYARRAY	Priv	A	test	
(1, 1)	N	1	(1.00000000)
(1, 2)	L	.F.		
(1, 3)	N	2	(2.00000000)
(1, 4)	N	3	(3.00000000)
(2, 1)	L	.F.		
(2, 2)	L	.F.		
(2, 3)	L	.F.		
(2, 4)	L	.F.		
(3, 1)	N	5	(5.00000000)
(3, 2)	L	.F.		
(3, 3)	N	6	(6.00000000)
(3, 4)	N	7	(7.00000000)
(4, 1)	N	9	(9.00000000)
(4, 2)	L	.F.		
(4, 3)	N	10	(10.00000000)
(4, 4)	N	11	(11.00000000)_

Another way of looking at the results is as follows:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

BECOMES

1	.F.	2	3
.F.	.F.	.F.	.F.
5	.F.	6	7
9	.F.	10	11

Returns

Numeric: 1 for a successful insert.

Note

Specifying the third parameter with a one-dimensional array will generate an error.

If there is no third parameter when inserting into a two-dimensional array, then a row is inserted.

The array size is not increased. In a one-dimensional array, the cells starting with nElementNumber are moved toward the end and a value of false (.F.) is inserted at that location. With a two dimensional array, the rows are pushed down and columns are pushed to the right starting at the nElementNumber row or column. That row or column is then filled with the value false (.F.).

See Also

[ACOPY\(\) Function](#)

[ADEL\(\) Function](#)

[ADIR\(\) Function](#)

[AELEMENT\(\) Function](#)

[ALEN\(\) Function](#)

[ASUBSCRIPT\(\) Function](#)

[DIMENSION Command](#)

AINSTANCE()

AINSTANCE(ArrayName, cClassName)

Remarks

Places instances of a class into a variable array and returns the number of elements created.

Parameters

ArrayName

The name of the array to which the instances will be written.

cClassName

The name of the class to search.

Example

```
CLEAR ALL
goObj1=CREATEOBJECT( 'TEXTBOX' )
goObj2=CREATEOBJECT( 'TEXTBOX' )

? AINSTANCE(gaTest, 'textbox')      && Returns 2
DISPLAY MEMORY LIKE gaTest        && Displays the two references
```

Returns

Numeric

Note

If the array name already exists and there are references returned, the array overwrites what's there and is given the corresponding number of elements. If the array name does not exist, it creates it.

Class elements assigned to arrays using CREATEOBJECT(), NEWOBJECT(), or linked to variables, such as in the DO FORM command are recognized.

See Also

ADD CLASS Command

AMEMBERS() Function

CREATE CLASS Command

CREATE CLASSLIB Command

CREATEOBJECT() Function

DEFINE CLASS Command

DO FORM

ALEN()

ALEN(ArrayName [,nArrayAttribute])

Remarks

Returns the number of elements, rows, or columns in an array.

Parameters

ArrayName

Name of the array you wish to check.

nArrayAttributes

Numeric expression that is used to determine if ALEN() returns the number of elements, rows, or columns.

Value	Description
0	Returns the number of elements. This is the default and the same as if you did not specify an attribute.
1	Returns the number of rows or elements.
2	Returns the number of columns.

Example

```
DIMENSION aTest[5,7]
? ALEN(aTest)      && Returns 35
? ALEN(aTest,1)    && Returns 5
? ALEN(aTest,2)    && Returns 7
```

Returns

Numeric

Note

If you only provide the array name, ALEN() returns the number of elements. Specifying the nArrayAttribute of 2 on a one-dimensional array returns 0.

See Also

[ADEL\(\) Function](#)

[ADIR\(\) Function](#)

[AELEMENT\(\) Function](#)

[AINS\(\) Function](#)

[ASUBSCRIPT\(\) Function](#)

[DIMENSION Command](#)

ALIAS()

ALIAS([nWorkArea | cAlias])

Remarks

Returns the table alias of the current or specified work area.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cAlias

Character expression indicating the alias to check.

Example

Note

If a numeric workarea value is provided, it will check the indicated workarea in the current data session and return the alias of any open table or cursor, otherwise an empty string is returned.

If the character alias is provided, it will act similar to the USED() function and check to see if that table alias is open in the current data session, returning it if found, otherwise an empty string is returned.

See Also

[DBF\(\) Function](#)

[SELECT\(\) Function](#)

[USE Command](#)

[SQL Class](#)

[Alias Property](#)

ALINES()

ALINES(ArrayName, cExpression [, nFlags][, cParseChar1 [, cParseChar2 [, ...]]])

Remarks

Copies lines in a character expression or memo field into an array.

Parameters

ArrayName

Name of the array to create that holds the individual lines parsed from cExpression.

cExpression

The character expression or field to parse.

nFlags

Numeric expression alters how the ALINES() function works and are described by the following table.

Bit	Additive Value	Description
0	1	(Default) Removes leading and trailing spaces and any trailing zeros (0), parse is case-sensitive, all elements are included, even if empty, does not include the parse characters in the results, does not include the last line if empty.
1	2	Include the last element in the array even if it is empty.
2	4	Do not include empty elements.
3	8	Specifies case-insensitive parse character searches.
4	16	Include the parsing characters in the results.

[, cParseChar1 [, cParseChar2 [, ...]]]

Specifies one or more parse characters used to break the data into separate lines. If no parse characters are included, then CHR(13) is the default.

Example

```
* A simple demonstration of ALINES()
SET CONSOLE ON      && Open the default console window
ACTIVATE CONSOLE    && Activate the default console window
CLEAR ALL
CLEAR
lcString="Fourscore and seven years ago our fathers brought forth,"+CHR(13)
        +"on this continent, a new nation, conceived in liberty, and"+CHR(13)
        +"dedicated to the proposition that all men are created equal."+CHR(13)
        +"Now we are engaged in a great civil war, testing whether that"+CHR(13)
        +"nation, or any nation so conceived, and so dedicated, can "+CHR(13)
        +"long endure."

? ALINES(aGettysburg1,lcString)          && Displays 6

? ALINES(aGettysburg2,lcString,0,',','.') && Displays 10

SET ALTERNATE TO GETTY.TXT
SET ALTERNATE ON
LIST MEMORY LIKE aGettysburg1
?
?
LIST MEMORY LIKE aGettysburg2
SET ALTERNATE OFF
SET ALTERNATE TO
```

Returns

Numeric

Note

If the array name does not exist, it will be created. If it does exist, it will be overwritten and given the corresponding number of elements.

In Windows, most text files terminate lines with the CHR(10)+CHR(13) character pair. If you do not want to include the CHR(10) in the results, you should use the TRANCHAR() command to first remove the unwanted characters.

See Also

[MEMLINES\(\) Function](#)

[ATLINE\(\) Function](#)

[CHRTRAN\(\) Function](#)

[SET ALTERNATE Command](#)

[SET MEMOWIDTH Command](#)

[STRTRAN\(\) Function](#)

[How JAXBase Deals with Arrays](#)

ALLTRIM()

ALLTRIM(cExpression, [, nFlags][, cParseChar1 [, cParseChar2 [, ...]]])

Remarks

Removes leading and trailing characters matching default or provided parameters.

Parameters

cExpression

Character or Binary Character expression that is to be trimmed.

nFlags – May be omitted

0 – (default) Case sensitive comparison of parse characters.

1 – Case insensitive comparison of parse characters.

2 – Remove only the first leading and trailing of each parse expression (case sensitive)

3 – Remove only the first leading and trailing of each parse expression (case insensitive)

cParseChars1 [, cParseChars2 [, ...]]

Optional parameters specifying one or more-character strings to remove from the beginning and end of cExpression.

Example

```
SET CONSOLE ON
ACTIVATE CONSOLE
SET MEMO WIDTH TO 0 && Turns off memo width tracking
CLEAR ALL
CLEAR

cStr="INSERT INTO PAID (account,paid) VALUES (acctno, payment-balance*(interest-discount))"

ALINES(aParsed1,cStr," INTO ")
ALINES(aParsed2,aParsed1[2],16," ( ")

lcTable=alltrim(aParsed2[1]," ","(")
lcFields=alltrim(aParsed2[2],1,")","(", " ",",","values")
lcValues=alltrim(aParsed2[3],3,")")

* Displays the following
*      LCTABLE      Priv  C  "PAID"
*      LCFIELDS     PRIV  C  "account,paid"
*      LCVVALUES    PRIV  C  "acctno, payment-balance*(interest-discount)
disp memo like lc*
```

Returns

Character

Note

If the array name does not exist, it will be created. If it does exist, it will be overwritten and given the corresponding number of elements.

When used with binary values, such as Varbinary and Blob, ALINES() creates an array with character values, so care must be taken when using binary data.

If no parse characters are provided, then both CHR(0) and CHR(32) are assumed.

See Also

[ALINES\(\) Function](#)

[ATLINE\(\) Function](#)

[MEMLINES\(\) Function](#)

[SET MEMOWIDTH Command](#)

[JAXBase Character Functions](#)

ALOCFILE()

ALOCFILE(ArrayName, cFileName, lRecurse)

Remarks

Attempts to locate the file and return the absolute path with the file name(s) in an array.

Parameters

ArrayName

Name of array to create containing any files that match cFileName.

cFileName

Character expression with a file name.

lRecurse

Logical expression indicating that ALOCFILE() should recurse sub-folders.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

ALOCFILE(aFiles, "*.*")
DISPLAY MEMORY LIKE aFiles
```

Return Value

Numeric

Note

ALOCFILE() returns the number of elements created corresponding to the number of files found.

If cFileName does not have an absolute path, then ALOCFILE() will attempt to search based on what was given and use the JAXBase path list to continue the search.

Care must be taken if setting lRecurse is true (.T.) as this could cause ALOCFILE() to search a large number of folders and files, slowing down processing.

If ArrayName does not exist, it will be created. If it exists, it will be overwritten, and the number of elements will correspond to the number of files found. If no results were found, the ArrayName is not created or modified.

See Also

[ADIR\(\) Function](#)
[FILE\(\) Function](#)
[GETFILE\(\) Function](#)
[GETPICTURE\(\) Function](#)
[PUTFILE\(\) Function](#)
[SET PATH Command](#)

DRAFT

AMEMBERS()

AMEMBERS(ArrayName, oObjectName [, nArrayContentsID] [, cFlags])

Remarks

Places the names of properties, procedures, and member objects for an object into an array.

Parameters

ArrayName

Name of the array that will hold the names of the member properties, procedures, or objects.

oObjectName

Specifies the object you are interested in. It can be expression that evaluates to an object.

nArrayContentsID

Specifies what to place into the array based on the following table.

Value	Description
0	Place just the property names into the array. Omitting this parameter is the same as 0.
1	Place the properties and methods of the object along with member objects. The array will be two-dimensional with the first column holding the name. The second column will hold "Property", "Event", "Method", or "Object".
2	Returns a two-dimensional array with the name of all child objects in column one, the class name in column two, and base class name in column three.

cFlags

Modifies what is returned in the array based on the following flags.

Flag	Description
F	Add an extra column that lists the following attributes for each entry: P – Protected H – Hidden G – Public N – Native to the object U – User defined C – Changed since INIT I – Inherited R – Read Only

Example

```
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

lcForm=CREATEOBJECT( "FORM" )

AMEMBERS(lcArray, lcForm, 1)      && Just return property names
DISPLAY MEMORY LIKE lcArray      && Display the form properties
```

Returns

Numeric

Note

If the array name does not exist, it will be created. If the array name already exists, it will be overwritten and the corresponding number of rows created, or 0 if none were created. If no rows are created, the array will not be created, and if the array name already exists, it will not be changed.

See Also

[ADD CLASS Command](#)

[AINSTANCE\(\) Function](#)

[CREATE CLASS Command](#)

[CREATE CLASSLIB Command](#)

[CREATEOBJECT\(\) Function](#)

[DEFINE CLASS Command](#)

APRINTOBJECTS()

APRINTOBJECTS(cArrayName [,cExpr])

Remarks

Returns one or more printer objects to the indicated array which relates to available printers. If you provide a character expression as the second parameter, it will search for all printer names containing that string and return them. The function returns an integer indicating how many printer objects were returned.

Parameters

cArrayName

Name of the array to place the printer objects.

cExpr

Character expression which is used as a partial name search.

Example

```
* Return all printers with HP in the name
* to the GAPRNT array
lnPrinterCount=APRINTOBJECTS(gaPrnt, "HP")

* Return the list of printers found
FOR i=1 TO lnPrinterCount
    ? gaPrnt.Name
ENDFOR
```

Return Value

Numeric

Note

If the array name does not exist, it will be created. If the array name already exists, it will be overwritten and the corresponding number of rows created, or 0 if none were created. If no rows are created, the array will not be created, and if the array name already exists, it will not be changed.

When it successfully finds one or more printers, it will create the array. If the variable already exists, it will release it and make the array. If there are no printers that match, then the array is not created. If the variable already exists, it is not changed.

See Also

Printer Class

ASC()

ASC(cExpression)

Remarks

Returns the ASCII value of the first character of the character expression.

Parameters

cExpression

A character expression.

Example

```
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

lcStr="FOUR SCORE"

FOR I=1 TO LEN(lcStr)
    ? ASC(SUBSTR(lcStr,I,1))
ENDFOR
```

Note

ASC() will only return the ASCII value of the first character if given more than one.

See Also

[CHR\(\)](#) Function

[SUBSTR\(\)](#) Function

[JAXBase Character Functions](#)

ASCAN()

```
ASCAN(ArrayName, eExpression [,nStartElement [,nElementsSearched [,nSearchColumn [,nFlags ]]]])
```

Remarks

Searches an array for an element matching the expression.

Parameters

Array Name

Name of the array to search.

eExpression

The expression to search for in the array.

nStartElement

Starting element for the search. If omitted, the entire array is searched. Specifying a 0 or negative number is the same as omitting the value.

nElementsSearched

Specifies how many elements to search. If you don't include nStartElement and nElementSearched, the search begins with the first array element and continues to the last array element if you do not specify nSearchColumn. Specifying 0 or a negative number is the same as omitting the value.

nSearchColumn

Specifies if only a column should be searched. Using 0 or a negative number will cause the whole array to be searched.

nFlags

nFlag	Bit	Description
0	0000	Case Sensitive; Exact OFF; Return element number
1	0001	Case Insensitive; Exact OFF; Return element number
2	0010	Case Sensitive; Exact ON; Return element number
3	0011	Case Insensitive; Exact ON; Return element number
4	0100	Case Sensitive; Exact OFF; Return row number
5	0101	Case Insensitive; Exact OFF; Return row number
6	0110	Case Sensitive; Exact ON; Return row number
7	0111	Case Insensitive; Exact ON; Return row number

Example

```
* Demonstrate ASCAN()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
COPY ALL TO ARRAY laCounties FOR STATE="CT"

* The following all return the
* array element for Hartfield, CT
? ASCAN(laCounties,"Hart")
? ASCAN(laCounties,"HART",0,0,1)
? ASCAN(laCounties,"Hartfield",0,0,2)
? ASCAN(laCounties,"HARTFIELD",0,0,3)
```

Return Value

Numeric

Note

If a match is found, ASCAN() returns the number of the element or row containing the expression. If a match cannot be found, ASCAN() returns 0.

Unless you specify a search column, nFlag 4 – 7 does not as useful as flags 0 - 3.

When searching, if using EXACT ON with nFlag, the match must be exact with all characters matching. With EXACT OFF, then as long as the element matches to the end of the expression, you have a match.

You must provide all parameters in or order to use the nFlag parameter, but nStartElement and nElementsSearched can be 0 or a negative number, which is the same as you omitted them.

See Also

- [ADEL\(\)](#) Function
- [ADIR\(\)](#) Function
- [AELEMENT\(\)](#) Function
- [AFIELDS\(\)](#) Function
- [AINS\(\)](#) Function
- [ASORT\(\)](#) Function
- [ASUBSCRIPT\(\)](#) Function
- [DIMENSION](#) Command
- [SET EXACT](#) Command

ASELOBJ() – Under consideration

ASELOBJ(`ArrayName`, [1 | 2 | 3])

Status

Under consideration

Remarks

Parameters

Example

Return Value

Note

If the array name does not exist, it will be created. If the array name already exists, it will be overwritten and the corresponding number of rows created, or 0 if none were created. If no rows are created, the array will not be created, and if the array name already exists, it will not be changed.

See Also

ASESSIONS()

ASESSIONS(ArrayName)

Remarks

Returns an array of active data sessions IDs.

Parameters

ArrayName

Name of the array into which ASESSIONS() will place the session IDs. Returns the number of elements created, which will always be at least 1 since Data Session 1 cannot be closed.

Example

```
* Demonstrate ASCAN()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

SELECT DATASESSION 2
SELECT DATASESSION 3

ASESSIONS(laDataSessions)
DISP MEMO LIKE laDataSessions

?
CLOSE DATASESSION 3

ASESSIONS(laDataSessions)
DISP MEMO LIKE laDataSessions
```

Return Value

Numeric

Note

If the array name does not exist, it will be created. If the array name already exists, it will be overwritten and the corresponding number of rows created, or 0 if none were created. If no rows are created, the array will not be created, and if the array name already exists, it will not be changed.

See Also

[SELECT DATASESSION Command](#)

[CLOSE DATASESSION Command](#)

[JAXBase Data Sessions](#)

ASIN()

ASIN(nExpression)

Remarks

Returns in radians the arc sine of a numeric expression.

Parameters

nExpression

A numeric expression ranging from +1 to -1.

Example

```
* Demonstrate ASCAN( )
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
```

Return Value

Numeric

Note

If the array name does not exist, it will be created. If the array name already exists, it will be overwritten and the corresponding number of rows created, or 0 if none were created. If no rows are created, the array will not be created, and if the array name already exists, it will not be changed.

The value ASIN() returns can range from $-\pi/2$ through $+\pi/2$ (-1.57079 to 1.57079).

The number of decimal places in the display of the result can be specified with SET DECIMALS.

Use RTOD() to convert radians to degrees.

See Also

RTOD() Function

SET DECIMALS Command

SIN() Function

ASORT()

```
ASORT(ArrayName [, nStartElement [, nNumberSorted [, nSortOrder [, nFlags]]])
```

Remarks

Sorts elements in an array in ascending or descending order. A return value of 1 means a successful sort, while -1 means a problem was encountered.

Parameters

ArrayName

Name of the array that will be sorted.

nStartElement

Numeric expression indicating starting element. If omitted, 0, or negative in value, then the sort starts at the first element. If the array is one-dimensional then nStartElement indicates the starting element. If the array is two-dimensional, then the starting element indicates the row and column based on how JAXBase works with arrays (see How JAXBase Deals with Arrays).

nNumberSorted

Specifies the number of elements to sort in a one-dimensional array or the number of rows to sort in a two-dimensional array. If 0 or a negative number, all elements or rows are sorted starting at nStartElement.

nSortOrder

Numeric expression for sort order where omitted, 0 or negative is ascending and if any positive value the sort order is descending.

nFlags

A numeric expression indicating case sensitivity where if omitted, 0 or negative means sorting is case-sensitive while a positive value indicates that case insensitive sorting is used.

Example

```
* Demonstrate ASCAN( )
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

DIMENSION laToSort[7]
FOR i = 1 TO 7
    laToSort[i]=INT(RAND()*1000)
ENDFOR

? "Starting values"
DISPLAY MEMORY LIKE laToSort

? "After sorting elements 2 through 6 in descending order"
ASORT(laToSort, 2, 5, 1)
DISPLAY MEMORY LIKE laToSort
```

Return Value

Numeric

Note

All elements involved in the sort must be of the same data type.

One-dimensional arrays or sorted by their elements. Two-dimensional arrays are sorted by their rows.

When a two-dimensional array is sorted, the rows are changed to align with the sorted column.

See Also

[ACOPY\(\) Function](#)

[ADEL\(\) Function](#)

[ADIR\(\) Function](#)

[AELEMENT\(\) Function](#)

[AFIELDS\(\) Function](#)

[AINS\(\) Function](#)

[ALEN\(\) Function](#)

[ASCAN\(\) Function](#)

[How JAXBase Deals with Arrays](#)

ASTACKINFO()

ASTACKINFO(ArrayName)

Remarks

Creates an array that contains information on the current call stack. Returns the number of rows created.

Parameters

ArrayName

Name of the array that will contain the information in the following column order.

Column	Description
1	Call stack level (starts at 1)
2	Program name
3	Module or Object name
4	Module or Object source filename
5	Line number in the source file
6	Source line contents (if available)

Example

```
* Demonstrate ATACKINFO()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

DO PROC1
RETURN

PROCEDURE PROC1
DO PROC2
RETURN

PROCEDURE PROC3
ASTACKINFO(laStack)
DISPLAY MEMO LIKE laStack
RETURN
```

Return Value

Numeric

Note

If the array name does not exist, it will be created. If the array name already exists, it will be overwritten and the corresponding number of rows created, or 0 if none were created. If no rows are created, the array will not be created, and if the array name already exists, it will not be changed.

ASTACKINFO() only populates column six if the source is available.

Using the second and fourth column, you can find out if the current program is a child of a program or object. A program will have a Fully Qualified File Name (FQFN) in column four. If JAXBase is unable to retrieve this information, it will be blank.

See Also

[LINENO\(\) Function](#)
[PROCEDURE Command](#)
[PROGRAM\(\) Function](#)

ASUBSCRIPT()

ASUBSCRIPT(ArrayName, nElementNumber, nSubscript)

Remarks

Returns the row or column number of the indicated element.

Parameters

ArrayName

Name of the array in question.

nElementNumber

Numeric expression indicating which element to test.

nSubscript

Numeric expression where 1 is for row and 2 is for column.

If the array is one dimensional and nSubscript is 1, then 1 is always returned, while a value of 2 for nSubscript will return the nElement number as single-dimension arrays are treated as two-dimensional with one row.

Example

```
* Demonstrate ASUBSCRIPT()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

CLEAR
DIMENSION laToSort[7]

? ASUBSCRIPT[laToSort,5,1]  && Returns 5
? ASUBSCRIPT[laToSort,5,2]  && Returns 0

DIMENSION laToSort[7,3]
? ASUBSCRIPT[laToSort,15,1] && Returns 3
? ASUBSCRIPT[laToSort,15,2] && Returns 1
```

Return Value

Numeric

Note

Two-dimensional arrays can be represented using [row,column] notation or [element] notation. Please see How JAXBase Deals with Arrays for more information.

If you give an nElementNumber that indicates a position outside of the array, an error will be generated.

See Also

ADEL() Function
ADIR() Function
AELEMENT() Function
AFIELDS() Function
AINS() Function
ALEN() Function
ASCAN() Function
ASORT() Function
DIMENSION Command
DISPLAY MEMORY Command
How JAXBase Deals with Arrays

DRAFT

AT()

```
AT(cSearchExpression, cExpressionSearched [, nOccurrence])
```

Remarks

Search a character expression or memo field for the occurrence of another character expression.

Parameters

cSearchExpression

The character expression to be used in the search.

cExpressionToSearch

The character expression to be used in the search. The expression may be a character field or memo field.

nOccurrence

Which occurrence to return. If omitted, 1 is assumed. A value less than 1 always returns 0.

Example

```
* Demonstrate ASCAN()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

aStr="FOUR SCORE"
? AT("O",aStr)           && Returns 2
? AT("O",aStr,2)         && Returns 8
? AT("s",aStr)           && Returns 0
```

Return Value

Numeric

Note

AT() is case sensitive. Use ATC to perform a case insensitive search.

If the indicated cSearchExpression or occurrence of same is not found then 0 is returned.

The value returned indicates the starting position of the first character of the matched cSearchExpression.

See Also

[AT_C\(\) Function](#)
[ATCC\(\) Function](#)
[ATLINE\(\) Function](#)
[LEFT\(\) Function](#)
[RAT\(\) Function](#)
[RATLINE\(\) Function](#)
[RIGHT\(\) Function](#)
[SUBSTR\(\) Function](#)
[SUBSTRC\(\) Function](#)
[STREXTRACT\(\) Function](#)
[\\$ Operator](#)
[OCCURS\(\) Function](#)
[INLIST\(\) Function](#)

AT_C()

```
AT_C(cSearchExpression, cExpressionSearched [, nOccurrence])
```

Status

Double-byte string handling is slated for Version 2

DRAFT

ATC()

ATC(nExpression)

Remarks

Search a character expression or memo field for the occurrence of another character expression.

Parameters

cSearchExpression

The character expression to be used in the search.

cExpressionToSearch

The character expression to be used in the search. The expression may be a character field or memo field.

nOccurrence

Which occurrence to return. If omitted, 1 is assumed. A value less than 1 always returns 0.

Example

```
* Demonstrate ASCAN()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

aStr="FOUR SCORE"
? AT("O",aStr)           && Returns 2
? AT("o",aStr,2)         && Returns 8
? AT("s",aStr)           && Returns 6
```

Return Value

Numeric

Note

ATC() is case insensitive. Use AT() to perform a case sensitive search.

If the indicated cSearchExpression or occurrence of same is not found, then 0 is returned.

The value returned indicates the starting position of the first character of the matched cSearchExpression.

See Also

[AT\(\) Function](#)
[AT_C\(\) Function](#)
[ATLINE\(\) Function](#)
[LEFT\(\) Function](#)
[RAT\(\) Function](#)
[RATLINE\(\) Function](#)
[RIGHT\(\) Function](#)
[SUBSTR\(\) Function](#)
[SUBSTRC\(\) Function](#)
[STREXTRACT\(\) Function](#)
[\\$ Operator](#)
[OCCURS\(\) Function](#)
[INLIST\(\) Function](#)

DRAFT

ATCC()

`ATCC(cSearchExpression, cExpressionSearched [, nOccurrence])`

Status

Double-byte string handling is slated for Version 2

DRAFT

ATCLINE()

ATCLINE(cSearchExpression, cExpressionToSearch, [,nOccurrence] [, cParseCharacter])

Remarks

Search an expression for a matching expression returning which line it was found.

Parameters

cSearchExpression

The character expression to be used in the search.

cExpressionToSearch

The character expression to be used in the search. The expression may be a character field or memo field.

nOccurrence

Which occurrence to return. If omitted, 1 is assumed. A value less than 1 always returns 0.

cParseCharacter

The character to use to terminate a line, such as CHR(13). If omitted, then the SET MEMOWIDTH value is used. If using SET MEMOWIDTH and its value is set to 0 then 1 is always returned for a match.

Example

```
* Demonstrate ATCLINE()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

lcString="Fourscore and seven years ago our fathers brought forth,"+CHR(13)
      +"on this continent, a new nation, conceived in liberty, and"+CHR(13)
      +"dedicated to the proposition that all men are created equal."+CHR(13)
      +"Now we are engaged in a great civil war, testing whether that"+CHR(13)
      +"nation, or any nation so conceived, and so dedicated, can "+CHR(13)
      +"long endure."

? ATCLINE("and",lcString,1,chr(13))      && Returns 1
? ATCLINE("and",lcString,2,chr(13))      && Returns 2
? ATCLINE("now",lcString,1,chr(13))      && Returns 4
```

Return Value

Numeric

Note

ATCLINE() is a case insensitive search.

If the search is successful, the line where the match was found is returned, otherwise 0 is returned.

If using SET MEMOWIDTH and its value is set to 0 then 1 is always returned for a match.

See Also

\$ Operator
AT() Function
ATC() Function
ATLINE() Function
INLIST() Function
LEFT() Function
MLINE() Function
OCCURS() Function
RAT() Function
RATLINE() Function
RIGHT() Function
SUBSTR() Function
SET MEMOWIDTH Command

ATLINE()

```
ATLINE(cSearchExpression, cExpressionToSearch, [,nOccurrence] [, cParseCharacter])
```

Remarks

Search an expression for a matching expression returning which line it was found.

Parameters

cSearchExpression

The character expression to be used in the search.

cExpressionToSearch

The character expression to be used in the search. The expression may be a character field or memo field.

nOccurrence

Which occurrence to return. If omitted, 1 is assumed. A value less than 1 always returns 0.

cParseCharacter

The character to use to terminate a line, such as CHR(13). If omitted, then the SET MEMOWIDTH value is used. If using SET MEMOWIDTH and its value is set to 0 then 1 is always returned for a match.

Example

```
* Demonstrate ATLINE()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

lcString="Fourscore and seven years ago our fathers brought forth,"+CHR(13)
      +"on this continent, a new nation, conceived in liberty, and"+CHR(13)
      +"dedicated to the proposition that all men are created equal."+CHR(13)
      +"Now we are engaged in a great civil war, testing whether that"+CHR(13)
      +"nation, or any nation so conceived, and so dedicated, can "+CHR(13)
      +"long endure.

? ATLINE("and",lcString,1,chr(13))      && Returns 1
? ATLINE("and",lcString,2,chr(13))      && Returns 2
? ATLINE("now",lcString,1,chr(13))      && Returns 0
```

Return Value

Numeric

Note

ATLINE() is a case sensitive search.

If the search is successful, the line where the match was found is returned, otherwise 0 is returned.

If using SET MEMOWIDTH and its value is set to 0 then 1 is always returned for a match.

See Also

\$ Operator
AT() Function
ATC() Function
ATCLINE() Function
INLIST() Function
LEFT() Function
MLINE() Function
OCCURS() Function
RAT() Function
RATLINE() Function
RIGHT() Function
SUBSTR() Function
SET MEMOWIDTH Command

ATN2()

ATN2(nYCoordinate, nXCoordinate)

Remarks

Returns the arc tangent in all four quadrants from specified values.

Parameters

nYCoordinate

Numeric expression of the Y coordinate.

nXCoordinate

Numeric expression of the X coordinate.

Example

```
* Demonstrate ASCAN()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? PI()                                && Displays 3.14
? ATN(0,-1)                            && Displays 3.14

gnX=COS(PI())
gnY=SIN(PI())

? ATN2(gnY,gnX)                        && Displays 3.14
? ATN2(gnY,gnX)/PI()                  && Displays 1
```

Return Value

Numeric

Note

ATN2() returns the angle in radians between the line $y=0$ and the line connecting the given coordinates and the origin $(0,0)$. Use RTOD() to convert to degrees.

ATN() returns a value between $\pi/2$ and $-\pi/2$.

See Also

ATAN() Function

RTOD() Function

SET DECIMALS Command

TAN() Function

ATOokens()

```
ATOokens(ArrayName, cStringToSearch [, cLeftBound, cRightBound [,lIncludeBounds]])
```

Remarks

Returns a list of tokens that appear between two different bounding character strings.

Parameters

ArrayName

Name of array to place the list of tokens.

cStringToSearch

Character based expression or field to search.

cLeftBound

Character expression containing the left boundary. Default is "<<".

cRightBound

Character expression containing the right boundary. Default is ">>".

lIncludeBounds

Logical flag indicating whether to include the bounding strings in the results. Default is true (.T.).

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
TEXT
Dear <<Name>>,
```

I sent you a bill for services rendered on <<Date1>> for an amount of <<Balance>> and have yet to receive payment. It is now <<days>> days overdue.

Please bring your account up to date by <<Date2>> or I shall be forced to seek legal remedies.

Thank you,

```
John Smith's Services
ENDTEXT
```

```
? ATOKENS(aToks)
```

```
* Array will contain four elements with the following values
* <<Date1>>, <<Balance>>, <<Days>>, <<Date2>>
DISPLAY MEMORY LIKE aToks
```

Return Value

Numeric

Note

ATOKENS() returns the number of elements returned in the array.

This command is helpful for creating your own mail merge logic as a preprocessor or complete replacement to the built in JAXBase features.

If the array name does not exist, it will be created. If the array name exists, it will be overwritten with the corresponding number of elements. If there are no results, the array name is not created or modified.

The character expressions cLeftBound and cRightBound can be any number of characters, but they must be different and ATOKENS() uses them as case insensitive boundaries.

cStringToSerch is unaffected by ATOKENS().

See Also

STREXTRACT() Function

TEXT / ENDTEXT Command

Merging Text in JAXBase

AUSED()

```
AUSED(ArrayName [, nDataSessionNumber [, cTableName]])
```

Remarks

Places table aliases and work areas for a data session into an array. Returns the number of rows created.

Parameters

ArrayName

Name of the array that will hold the returned results.

nDataSessionNumber

Numeric expression for the data session to interrogate.

cTableName

Character expression for the table name to use in the search.

Valid expressions are file name stem, and a Fully Qualified File Name (FQFN).

Example

```
* Demonstrate ASCAN()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
SELECT FROM (_JAXFolder+"data\counties") WHERE State="CT" ALIAS CT
SELECT FROM (_JAXFolder+"data\counties") WHERE State="VA" ALIAS VA
SELECT FROM (_JAXFolder+"data\counties") WHERE State="CT" ALIAS SC
USE (_JAXFolder+"data\phonebook")

AUSED(laMatches, "counties")
DISPLAY MEMORY LIKE laMatches
```

Return Value

Numeric

Note

If the array name does not exist, it will be created. If the array name already exists, it will be overwritten and the corresponding number of rows created, or 0 if none were created. If no rows are created, the array will not be created, and if the array name already exists, it will not be changed.

AUSED returns an array with a row for each match where the first column is the table alias, and the second column is the work area it is in.

If a FQFN is given, then only tables that are a match to that exact FQFN are returned. If just a file stem, then any match for files or SQL table names are returned.

See Also

ALIAS() Function

JUSTSTEM() Function

SET DATASESSION Command

USE Command

DRAFT

AWORKAREAS()

AWORKAREAS(ArrayName [,nDataSessionID])

Remarks

Returns a list of all active work areas in a two-dimensional array that has the work area ID in column 1 and the alias name or empty character string, if no table or cursor is open, in column 2. The number of rows created is returned.

Parameters

ArrayName

nDataSessionID

Numeric Expression of what data session to search. If omitted, the current data session is used.

Example

```
* Demonstrate the AWORKAREAS() Function
CLOSE ALL
CLEAR ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
SELECT 1
SELECT 2
? AWORKAREAS(laWorkAreas)      && Displays 2
DISPLAY MEMORY LIKE laWorkAreas && List the results
```

Return Value

Numeric

Note

If the array name does not exist, it is created. If it does exist, it will be overwritten and the corresponding array elements created for each entry. There will always be at least 1 entry as work area 1 is always created when a data session is initialized.

If you specify a data session in parameter 2 that does not exist, an error will be generated.

See Also

[ADATASESSIONS\(\) Function](#)

[ALIAS\(\) Function](#)

[SELECT Command](#)

[SELECT DATASESSION Command](#)

[SELECT\(\) Function](#)

BETWEEN()

BETWEEN(eTestValue, eLowValue, eHighValue)

Remarks

Tests whether eTestValue is between eLowValue and eHighValue, inclusive. Returns a logical true (.T.) or false (.F.).

Parameters

eTestValue

An expression representing the value to test.

eLowValue

The acceptable low end of the range to check.

eHighValue

The acceptable high end of the range to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
* Returns .T.
? BETWEEN(5,3,10)
? BETWEEN("C","A","Z")
? BETWEEN({01/15/2023},{10/01/2022},{5/16/2023 13:01:00})
```

```
* Returns .F.
? BETWEEN(5,6,10)
? BETWEEN("C","a","z")
```

Return Value

Logical true (.T.) if the test succeeds, otherwise logical false (.F.).

Note

All values must be of the same data type.

Dates and DateTimes can be intermingled as a DATE is actually stored as a DATETIME with the time portion set to 00:00:00.

See Also

[INLIST\(\) Function](#)

[MAX\(\) Function](#)

[MIN\(\) Function](#)

BINDEVENT()

```
BINDEVENT(oEventSource, cEvent, oEventHandler, cDelegate [, nFlags])
```

```
BINDEVENT(hWnd | 0, nMessage, oEventHandler, cDelegate [, nFlags])
```

Remarks

Status

Slated for Version 2

DRAFT

BINTOC()

BINTOC(nExpression [, eFlags])

Remarks

Converts a numeric value to a binary character string.

Parameters

nExpression

Numeric expression to convert.

cFlags

Expressed as a 1 to 4 byte string the first byte is 1, 2, 4, or 8, indicating length of the character string returned. The second character is I indicating integer, D for decimal, and U for unsigned integer. An integer can be any length, but a Decimal is only 4 bytes (Float) or 8 bytes (Currency/Double). The optional third character will be R indicating the string is to be reversed.

If omitted, the result will be a 4 character, signed integer representation ("4I"). If the string is just a single character 1,2,4, or 8 then a signed integer value representation of nExprssion is returned.

If the value is larger than the specified results, an error is raised. The following table indicates expression ranges.

1	-127 to 127 or 0 to 255 if unsigned
2	-32,768 to 32,767 or 0 to 65,535 if unsigned
4	Approximately 1.5 E - 45 to 3.4 E + 38 for a decimal value or -2,147,486,648 to 2,187,483,647 for signed integer or 0 to 4,294,967,296 for unsigned integer
8	Approximately 2.2250738 E - 308 to 1.7976931 E + 308 for a decimal value or -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 for signed integer or 0 to 18,446,744,073,709,551,615 for unsigned integer

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? BINTOC(1,"1")
? BINTOC(255,"1S")
? BINTOC(124.06,"4F")
? BINTOC(124.06,"8F")
```

Return Value

Character

Note

When dealing with large numbers (4-byte Float, 8-byte Double, or Integers) you will lose precision when it returns scientific notation values.

JAXBase will protect integer values up to 72,057,594,037,927,936.

JAXBase will protect precision of 8-byte decimal values in the range of -549,755,813,999.9999 to 549,755,813,999.9999 and outside of that range it is subject to exponential notation and loss of information. Greater precision to the right of the decimal point reduces precision on the left.

See Also

[CTOBIN\(\) Function](#)
[CREATEBINARY\(\) Function](#)
[Double Field Type](#)
[Float Field Type](#)
[Integer Field Type](#)

BITAND()

BITAND(nExpression1, nExpression2)

Remarks

Performs a bitwise AND operation on two integer values and returns the result.

Parameters

nExpression1, nExpression2

Integer values used in the BITAND expression.

Example

```
* Demonstrate BITAND( )
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=43          && 00101011
C=226         && 11100010

? BITAND(A,C) && Returns 34
```

Return Value

Numeric

Note

If either numeric expression is not an integer, it will be converted.

The following table shows the result of a bitwise AND operation between two bits

Bit 1	Bit 2	AND Result
0	0	0
0	1	0
1	0	0
1	1	1

See Also

[BITCLEAR\(\) Function](#)
[BITLSHIFT\(\) Function](#)
[BITNOT\(\) Function](#)
[BITOR\(\) Function](#)
[BITRSHIFT\(\) Function](#)
[BITSET\(\) Function](#)
[BITTEST\(\) Function](#)
[BITXOR\(\) Function](#)

BITCLEAR()

BITCLEAR(nExpression1, nExpression2)

Remarks

Clears the bit position nNumericExpression2 in the integer value of nNumericExpression1.

Parameters

nExpression1

Numeric expression that will have a bit cleared.

nExpression2

Numeric expression that indicates which bit is to be cleared.

Example

```
* Demonstrate BITCLEAR()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=255
B=2          && 0 is first bit position
? BITCLEAR(A,B)      && Returns 251
```

Return Value

Numeric

Note

BITCLEAR() operations will only support a nNumericExpression2 up to 31
If either numeric expression is not an integer, it will be converted.

See Also

- BITCLEAR() Function
- BITLSHIFT() Function
- BITNOT() Function
- BITOR() Function
- BITRSHIFT() Function
- BITSET() Function
- BITTEST() Function
- BITXOR() Function

BITLSHIFT()

BITLSHIFT(nExpression1, nExpression2)

Remarks

Moves the bits in nExpression1 left by nExpression2 positions.

Parameters

nExpression1

Numeric Expression to alter.

nExpression2

Numeric expression indicating how many bits to move.

Example

```
* Demonstrate BITLSHIFT( )
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=255
B=2          && 0 is first bit position
? BITLSHIFT(A,B)    && Returns 1020
```

Return Value

Numeric

Note

BITLSHIFT() operations will only support a nNumericExpression2 up to 31
If either numeric expression is not an integer, it will be converted.

See Also

- [BITCLEAR\(\) Function](#)
- [BITLSHIFT\(\) Function](#)
- [BITNOT\(\) Function](#)
- [BITOR\(\) Function](#)
- [BITRSHIFT\(\) Function](#)
- [BITSET\(\) Function](#)
- [BITTEST\(\) Function](#)
- [BITXOR\(\) Function](#)

BITNOT()

BITNOT(nExpression)

Remarks

Performs a bitwise NOT operation on the nNumericExpression.

Parameters

nNumericExpression

Numeric expression used in the NOT operation.

Example

```
* Demonstrate BITNOT()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=15
? BITNOT(A,B)      && Returns -16
```

Return Value

Numeric

Note

If nNumericExpression is not an integer value, it will be converted.

See Also

[BITCLEAR\(\) Function](#)
[BITLSHIFT\(\) Function](#)
[BITNOT\(\) Function](#)
[BITOR\(\) Function](#)
[BITRSHIFT\(\) Function](#)
[BITSET\(\) Function](#)
[BITTEST\(\) Function](#)
[BITXOR\(\) Function](#)

BITOR()

BITOR(nExpression1, nExpression2)

Remarks

Perform a bitwise inclusive OR operation on two values.

Parameters

nExpression1, nExpression2

Numeric expressions to perform the OR operation.

Example

```
* Demonstrate BITOR()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=15
B=164
? BITNOT(A,B)      && Returns 175
```

Return Value

Numeric

Note

If either numeric expressions are not integers, they will be converted.

The following table shows the result of a bitwise OR operation between two bits

Bit 1	Bit 2	OR Result
0	0	0
0	1	1
1	0	1
1	1	1

See Also

[BITCLEAR\(\) Function](#)

[BITLSHIFT\(\) Function](#)

[BITNOT\(\) Function](#)

[BITOR\(\) Function](#)

[BITRSHIFT\(\) Function](#)

[BITSET\(\) Function](#)

[BITTEST\(\) Function](#)

[BITXOR\(\) Function](#)

BITRSHIFT()

BITRSHIFT(nExpression1, nExpression2)

Remarks

Moves the bits in nExpression1 right by nExpression2 positions.

Parameters

nExpression1

Numeric Expression to alter.

nExpression2

Numeric expression indicating how many bits to move.

Example

```
* Demonstrate BITLSHIFT( )
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=255
B=2          && 0 is first bit position
? BITRSHIFT(A,B)      && Returns 63
```

Return Value

Numeric

Note

BITRSHIFT() operations will only support a nNumericExpression2 up to 31
If either numeric expression is not an integer, it will be converted.

See Also

[BITCLEAR\(\) Function](#)
[BITLSHIFT\(\) Function](#)
[BITNOT\(\) Function](#)
[BITOR\(\) Function](#)
[BITRSHIFT\(\) Function](#)
[BITSET\(\) Function](#)
[BITTEST\(\) Function](#)
[BITXOR\(\) Function](#)

BITSET()

BITSET(nExpression1, nExpression2)

Remarks

Sets a bit to 1 in a value.

Parameters

nExpression1

Value to alter.

nExpression2

Expression of bit to set to 1.

Example

```
* Demonstrate BITSET
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=15
B=4
? BITNOT(A,B)      && Returns 31
```

Return Value

Numeric

Note

If either numeric expression is not an integer, it will be converted.

Valid bit values for nExpression2 are 0 to 31.

See Also

[BITCLEAR\(\) Function](#)

[BITLSHIFT\(\) Function](#)

[BITNOT\(\) Function](#)

[BITOR\(\) Function](#)

[BITRSHIFT\(\) Function](#)

[BITSET\(\) Function](#)

[BITTEST\(\) Function](#)

[BITXOR\(\) Function](#)

BITTEST()

BITTEST(nExpression1, nExpression2)

Remarks

Returns a logical True (.T.) value if the specified bit is set.

Parameters

nExpression1

Numeric expression to test.

nExpression2

Numeric expression of bit to test.

Example

```
* Demonstrate BITTEST( )
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

A=15
? BITTEST(A,0)      && Returns .T.
? BITTEST(A,1)      && Returns .T.
? BITTEST(A,2)      && Returns .T.
? BITTEST(A,3)      && Returns .T.
? BITTEST(A,4)      && Returns .F.
```

Return Value

Numeric

Note

If either numeric expression is not an integer, it will be converted.

The valid bit range for nExpression2 is 0 to 31.

See Also

- BITCLEAR() Function
- BITLSHIFT() Function
- BITNOT() Function
- BITOR() Function
- BITRSHIFT() Function
- BITSET() Function
- BITTEST() Function
- BITXOR() Function

BITXOR()

BITXOR(nExpression1, nExpression2)

Remarks

Perform a bit-wise exclusive OR operation on two values.

Parameters

nExpression1, nExpression2

Two values with which to perform the operation.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? BITXOR(255,15)      && Returns 240
? BITXOR(240,15)      && Returns 255
```

Return Value

Numeric

Note

If either numeric expression is not an integer, it will be converted.

The following table shows the result of a bitwise XOR operation between two bits

Bit 1	Bit 2	AND Result
0	0	0
0	1	1
1	0	1
1	1	0

See Also

[BITCLEAR\(\) Function](#)
[BITLSHIFT\(\) Function](#)
[BITNOT\(\) Function](#)
[BITOR\(\) Function](#)
[BITRSHIFT\(\) Function](#)
[BITSET\(\) Function](#)
[BITTEST\(\) Function](#)
[BITXOR\(\) Function](#)

BOF()

`BOF([nWorkArea | cTableAlias])`

Remarks

Returns logical .T. when record pointer is at the beginning of the table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate BOF()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
INDEX ON NAME TO COUNTIES_NAME

GOTO BOTTOM
? RECNO( ),BOF()

WHILE NOT BOF()
    Skip -1
ENDDO

? RECNO( ),BOF()
```

Return Value

Logical

Note

BOF() returns true (.T.) if you try to move the record pointer above the top record. If an index is in effect, the top record may not be the first record in the table. BOF() accurately represents the new indexed top of file.

See Also

[EOF\(\) Function](#)

[FEOF\(\) Function](#)

CANDIDATE()

```
CANDIDATE(nIndexNumber [,nWorkArea } cAlias])
```

Remarks

Returns a value of true (.T.) if the index is a candidate index.

Parameters

nIndexNumber

Numeric expression indicating the index to use. Any index in JAXBase can be marked as candidate indexes.

nWorkArea

Numeric expression indicating the work area.

cAlias

Character expression indicating the table alias to query.

Example

```
* Demonstrate BOF()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

Return Value

Logical

Note

If workarea and table alias are omitted, the current workarea is used.

If nIndexNumber is less than 1, an error is generated. If nIndexNumber goes beyond the scope of open indexes, it will return false (.F.).

See Also

INDEX Command

CAPSLOCK()

CAPSLOCK(lExpression)

Remarks

Returns current CAPS LOCK mode and optionally sets it.

Parameters

lExpression

Logical expression that turns CAPS LOCK on when true (.T.) and off when false (.F.).

Example

```
* Demonstrate CAPSLOCK()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
g1oldCapsLock=CAPSLOCK()
```

```
g1NewCapsLock=CAPSLOCK(!g1oldCapsLock)
```

```
? g1oldCapsLock, g1NewCapsLock
```

Return Value

Logical

Note

If you do not include an argument with CAPSLOCK(), it will simply return the CAPS LOCK state.

See Also

[INSMODE\(\) Function](#)

[NUMLOCK\(\) Function](#)

CDOW()

`CDOW(dExpression | tExpression)`

Remarks

Returns day of week from the provided Date or DateTime expression.

Parameters

dExpression

Date expression that is used to return the day of the week.

tExpression

DateTime expression that is used to return the day of the week.

Example

```
* Demonstrate DOW()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
SET STRICT DATE TO 0
ACTIVATE CONSOLE
CLEAR

? DOW({09/05/2025}) && Displays Friday
```

Return Value

Character

Note

CDOW() returns the proper name of the day of the week in question.

See Also

- DATE() Function
- DATETIME() Function
- DAY() Function
- DOW() Function
- SET FDOW Command
- SET FWEEK Command

CEILING()

CEILING(nExpression)

Remarks

Returns the next highest integer that is greater than or equal to nExpression.

Parameters

nExpression

Numeric expression used by CEILING() to get the next highest integer.

Example

```
* Demonstrate CEILING( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? CEILING(10)      && Displays 10
? CEILING(10.01)   && Displays 11
? CEILING(-10.1)   && Displays -10
```

Return Value

Numeric

Note

CEILING() rounds a number with a fractional decimal component to the next highest integer.

See Also

FLOOR() Function
ROUND() Function
Integer Field Type

CHR()

CHD(nExpression)

Remarks

Returns the character associated with the numeric expression.

Parameters

nExpression

Numeric expression in the range of 0 to 255.

Example

```
* Demonstrate CHR()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? CHR(49)      && Displays 1
? CHR(65)      && Displays A
? CHR(97)      && Displays a
```

Return Value

Character

Note

CHR() returns a single character representing the corresponding code from the codepage table. Prior to Version 2.0, JAXBase will use the current default codepage.

See Also

[ASC\(\) Function](#)

[INKEY\(\) Function](#)

[SET CODEPAGE Command](#)

CHRTRAN()

CHRTRAN(cExpression, cSearchExpression, cReplacementExpression)

Remarks

Replaces letters in cExpression based on cSearchExpression and cReplacementExpression.

Parameters

cExpression

Character expression that will be modified by the CHRTRAN() function.

cSearchExpression

Character expression holding the characters that will be modified.

cReplacementExpression

Character expression containing the characters that will replace the cSearchExpression characters in the cExpression.

Example

```
* Demonstrate CHRTRAN( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? CHRTRAN( "Moby Dick", "oI", "0!")      && Displays M0by D!ck
? CHRTRAN( ">    PASSWORD    <", " AO", "") && Displays >PSSWRD<
```

Return Value

Character

Note

cSearchExpression and cReplacementExpression are matched up character by character from left to right. If cReplacementExpression is not as long as cSearchExpression, then the characters in cSearchExpression that have no match are removed from cExpression.

See Also

[CHRTRANC\(\) Function](#)

[STRTRAN\(\) Function](#)

[Creating Character Expressions](#)

CHRTRANC()

`CHRTRAN(cExpression, cSearchExpression, cReplacementExpression)`

Status

Slated for Version 2.0

DRAFT

CMONTH()

CMONTH(dExpression | tExpression)

Remarks

Returns the name of the month for the specified date.

Parameters

dExpression

Date expression used to return the name of the month.

tExpression

DateTime expression used to return the name of the month.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET STRICTDATE TO 0
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? CMONTH({01/01/2025})      && Displays January
? CMONTH({07/01/2025})      && Displays July
```

Return Value

Character

Note

CMONTH() returns the name of the month as a string in proper noun format.

See Also

[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[DMY\(\) Function](#)
[MDY\(\) Function](#)
[MONTH\(\) Function](#)

COMMIT()

`COMMIT()`

Remarks

Attempts to commit the current SQL database transaction. Returns a logical true (.T.) if successful.

Return Value

Logical

Note

If there is no open transaction in the current workarea, an error will be generated. COMMIT() will work against a SQL table that was opened under an OPEN DATABASE or using a SQL class.

COMMIT() returns logical true (.T.) if successful and false (.F.) if not.

COMMIT() may encounter an error during the process, but it is recorded rather than interrupting the program flow. You can use the AERROR() or the SQL class Error() function to discover the reason for the failure.

Under most circumstances, a failed commit can be tried again if the problem is fixed. Otherwise, you need to ROLLBACK() the transaction.

If an ENDTRANSACTION is encountered and the transaction is still open, a COMMIT() will be attempted and if it fails, an error is generated.

See Also

BEGIN TRANSACTION Command
ENDTRANSACTION Command
OPEN DATABASE Command
ROLLBACK()
SQL Class

COMPILE()

COMPILE(cCodeBlock)

Remarks

Compiles JAXBase source code into object code that can be placed into a class method or event using the MEUPDATE method in most user classes.

Parameters

cCodeBlock

A character string that contains valid JAXBase commands.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

aStr=COMPILE( "DO TEST1" )

FOR I=1 to len(aStr)
    ?? ASC(SUBSTR(aStr,I,1))
ENDFOR
```

Note

JAXBase allows on-the-fly updates to most events and methods in most user classes. This is a drastic departure from most xBase dialects and is one of the many powerful features of JAXBase.

See Also

COMPILE Command
MEUPDATE method

COS()

cos(nExpression)

Remarks

Returns the cosine of a numeric expression.

Parameters

nExpression

Numeric expression used by the COS() function.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? COS(0)      && Displays 1.00
? COS(PI())   && Displays -1.00
? COS(45)     && Displays 0.53
? COS(1024)   && Displays 0.99
```

Return Value

Numeric

Note

COS() returns radians. Use DTOR() to convert an angle from degrees to radians and RTOD() to convert radians to degrees.

The number of decimal places returned is dictated by SET DECIMALS. The value COS() returns is between -1 and +1 inclusive.

See Also

- ACOS() Function
- DTOR() Function
- RTOD() Function
- SET DECIMALS Command
- SIN() Function
- TAN() Function

CP CONVERT()

Status

Slated for Version 2.0

DRAFT

CPCURRENT()

Status

Slated for Version 2.0

DRAFT

CPDBF()

Status

Slated for Version 2.0

DRAFT

CREATEBINARY()

Status

Slated for Version 2.0



CREATEOBJECT()

CREATEOBJECT(cClassName [,eParametersArray])

Remarks

Creates an object from a built in or user defined class definition.

Parameters

cClassName

Name of the built in JAXBase class or a user defined class.

eParametersArray

A two-dimensional array with the first column holding the property name and the second column holding the value to place into the property.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
aForm=CREATEOBJECT( "FORM" )  
? aForm.Name      && Displays FORM1
```

Return Value

Class Object

Note

Properties names in the array that do not exist will attempt to be added to the class. If a property cannot be added or updated, an error will be generated.

The value of empty string will default to the empty value of that property.

See Also

DEFINE CLASS Command
GETOBJECT() Function
NEWOBJECT() Function
RELEASE Command
SET CLASSLIB Command

CTOBIN()

`CTOBIN(cExpression [,cFlags])`

Remarks

Converts a binary encoded number into a numeric value.

Parameters

cExpression

The character expression to convert.

cFlags

If the character expression is an 8 byte expression, you can control what is returned using the following:

cFlags is used to control what kind of value is returned where the first character is 1, 2, 4, or 8, indicating the type of value returned as shown in the table below. The second character is I indicating integer, D for decimal, and U for unsigned integer. An integer can be any length, but a Decimal is only 4 bytes (Float) or 8 bytes (Currency/Double). The optional third character will be R indicating the original string is to be reversed before conversion.

If omitted, the result will be a signed integer representation ("4I").

If the original string is not 8 bytes in length, then cFlags is ignored.

1	-127 to 127 or 0 to 255 if unsigned
2	-32,768 to 32,767 or 0 to 65,535 if unsigned
4	Approximately 1.5 E - 45 to 3.4 E + 38 for a decimal value or -2,147,486,648 to 2,187,483,647 for signed integer or 0 to 4,294,967,296 for unsigned integer
8	Approximately 2.2250738 E – 308 to 1.7976931 E + 308 for a decimal value or -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 for signed integer or 0 to 18,446,744,073,709,551,615 for unsigned integer

Example

```
* Demonstrate CTOBIN()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
aStr=BINTOC(1024)  
? CTOBIN(aStr)
```

```
aStr=BINTOC(8675309,"8I")  
? CTOBIN(aStr,"4I")
```

```
aStr=BINTOC(10,"8D")  
? CTOBIN(astr,"1I")
```

Return Value

Numeric

Note

If cExpression or cFlags have invalid information, an error is generated.

See Also

[BINTOC\(\) Function](#)

CTOD()

`CTOD(cExpression)`

Remarks

Convert a character date expression to a Date data type.

Parameters

`cExpression`

Character expression in a recognized Date format. See How JAXBase Handles Dates.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
SET STRICTDATE TO 0
ACTIVATE CONSOLE
CLEAR

cDate="09/05/2025"
? CTOD(cDate)
? CMONTH(CTOD(cDate))
? CTOD("09-05-2025")
? CTOD("2025-09-05")

SET DATE TO BRITISH
? CMONTH(CTOD("05-09-2025"))
```

Return Value

Date

Note

If an invalid date expression is encountered, a blank date value { / / } is returned.

See Also

[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[CMONTH\(\) Function](#)
[SET DATE Command](#)
[SET STRICTDATE Command](#)

CTOT()

CTOT(cExpression)

Remarks

Convert a character date expression to a DateTime data type.

Parameters

cExpression

Character expression in a recognized DateTime format. See How JAXBase Handles Dates.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
SET STRICTDATE TO 0
SET HOURS TO 12
SET DATE TO AMERICAN
ACTIVATE CONSOLE
CLEAR

cDate="09/05/2025"
? CTOT(cDate)          && Displays 09-05-2025 12:00:00 AM
? CMONTH(CTOT(cDate)) && Displays September
? CTOT("09-05-2025 00:14:16") && Displays 09-05-2025 12:14:16 AM
? CTOT("2025-09-05 14:16:18") && Displays 09-05-2025 02:16:18 PM

SET DATE TO BRITISH
? CMONTH(CTOT("05-09-2025 12:05:00 AM")) && Displays September
```

Return Value

Date

Note

If an invalid date expression is encountered, a blank date value {::} is returned.

Times are expressed in 12 or 24 hour format, depending on the SET TIME command. However, if a DateTime expression does not end in AM or PM, the time portion is automatically considered to be 24 hour (military) format.

See Also

[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[CMONTH\(\) Function](#)
[SET DATE Command](#)
[SET STRICTDATE Command](#)

CURSORGETPROP()

Status

Slated for Version 2.0

DRAFT

CURSORSETPROP()

Status

Slated for Version 2.0

DRAFT

CURSORTOJSON()

CURSORTOJSON(nWorkArea | cAlias, cFile)

Status

Slated for Version 2.0

Remarks

Writes the records of a table or cursor to a JSON file.

Parameters

nWorkArea

Numeric expression that indicates which work area to use.

cAlias

Character expression that indicates which alias to use.

cFile

Character expression that holds the name of the file to create.

Return Value

Numeric

Note

CURSORTOJSON() returns the number of records written. If an error occurs during the process, the JSON file will not be created.

To limit the records sent to the JSON file, set a FILTER.

See Also

[CURSORTOXML\(\) Function](#)

[JSONTOCURSOR\(\) Function](#)

[SET FILTER Command](#)

[XMLTOCURSOR\(\) Function](#)

CURSORTOXML()

`CURSORTOXML(nWorkArea |cAlias, cFile[, nFlags])`

Status

Slated for Version 2.0

Remarks

Writes the records of a table or cursor to XML.

Parameters

nWorkArea

Numeric expression that indicates which work area to use.

cAlias

Character expression that indicates which alias to use.

cFile

Character expression that holds the name of the file to create.

nFlags

nFlag	Description
0	Write the data to the XML file (default)
1	Write the schema to the XML file
2	Write the schema and data to the XML file

Return Value

Numeric

Note

CURSORTOXML() returns the number of records written. If an error occurs during the process, the XML file will not be created.

To limit the records sent to the XML file, set a FILTER.

See Also

[CURSORTOJSON\(\) Function](#)

[JSONTOCURSOR\(\) Function](#)

[SET FILTER Command](#)

[XMLTOCURSOR\(\) Function](#)

CURVAL()

```
CURREC(nRecNo [, lRecStatus] [, nWorkArea | cAlias])
```

Remarks

Returns a record object directly from the current record in a JAXBase table.

Parameters

nRecNo

Numeric expression of the record to fetch. A record number of 0 indicates the current record.

lRecStatus

If set to a value of true (.T.), returns the following properties in the Record Object.

Name	Data Type	Description
_CURIDX	Integer	Value of the index in control. Use NDX() to get the name.
_DELETED	Logical	A value of true (.T.) indicates the record is marked for deletion
_RECNO	Integer	Absolute record number

nWorkArea

Numeric expression that indicates which work area to use.

cAlias

Character expression that indicates which alias to use.

Example

```
* Demonstrate CURVAL( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
GOTO TOP

SCATTER NAME oRec1          && Fetch the values of record 1
oRec2=CURVAL(100,.T.)       && Fetch the values of record 100
DISP MEMO LIKE orec*        && Display the two records
```

Return Value

JAXBase Record Object

Note

A JAXBase Record Object has properties named after the fields and reflect the current values based on the currently selected record. If the record is past the end of the file, an error is generated.

If lRecStatus is set to true (.T.) and there are fieldnames that match the property names that match the status properties, an error will be generated and an empty JAXBase Record Object will be returned.

If the work area or alias, the current workarea is used.

This command is typically used to quickly compare the current value of fields in a table to see if they have been altered in a multiuser environment.

nRecNo references is the absolute record number in the table and is unaffected by indexes. The value of 0 indicates the current record under the record pointer.

The CURVAL() command does not move the record pointer, so if the record pointer is set to record 10 and nRecNo is 100, it will fetch the values from record 100 without moving the record pointer.

See Also

EMPTY Class
EOF() Function
GOTO Command
SCATTER Command

DATE()

`DATE([nYear, nMonth, nDay])`

Remarks

Returns the operating system Date value or creates a Y2K compliant Date value.

Parameters

nYear

A numeric value between 1 and 9999.

nMonth

A numeric value between 1 and 12 for month.

nDay

A numeric value for the date between 1 and 31.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET DATE TO AMERICAN
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? DATE()           && Returns today's date
? Date(2005,12,31) && Returns 12/31/2005
```

Return Value

Date value type

Note

If the combined year, month, and day values do not combine into a valid date (example: 02/31/2025) then a blank date ({ / / }) is returned.

See Also

- CTOD() Function
- DATETIME() Function
- DTOC() Function
- SET CENTURY Command
- SET DATE Command

DATETIME()

DATETIME([nYear, nMonth, nDay [,nHours [,nMinutes , nSeconds]]])

Remarks

Returns the operating system DateTime value or creates a Y2K compliant DateTime value.

Parameters

nYear

A numeric expression that evaluates to 1 and 9999 representing year.

nMonth

A numeric expression that evaluates to 1 and 12 representing month.

nDay

A numeric expression that evaluates to 1 and 31 representing day of month.

nHours

A numeric expression that evaluates to 0 through 23 representing the hour in military time.

nMinutes

A numeric expression that evaluates to 0 through 59 representing minutes.

nSeconds

A numeric expression that evaluates to 0 through 59 representing seconds.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET DATE TO AMERICAN
SET HOUR TO 12
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? DATETIME()                                && Returns current DateTime from the operating system
? DATETIME(2005,12,31,23,59,59})  && Returns 12/31/2005 11:59:59 PM
```

Return Value

DateTime value type

Note

If the combined values do not correspond to a valid DateTime value, then a blank DateTime value ({}//:{})) is returned.

See Also

CTOT() Function
DATE() Function
DTOT() Function
HOUR() Function
SEC() Function
SECONDS() Function
SET DATE Command
SET HOUR Command
SET SECONDS Command
SET SYSFORMATS Command
TIME() Function
TTOC() Function
TTOD() Function

DRAFT

DAY()

DAY(dExpression | tExpression)

Remarks

Returns the numeric day of the month for a given Date or DateTime value.

Parameters

dExpression

Date expression used to return the name of the month.

tExpression

DateTime expression used to return the name of the month.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

? DAY({09/05/2025})	&& Displays 5
? DOW({09/05/2025})	&& Displays 6
? DAY({11/22/1963})	&& Displays 22
? DOW({11/22/1963})	&& Displays 6
? DAY({01-02-2001 00:00:01})	&& Displays 2
? DOW({01-02-2001 00:00:01})	&& Displays 3

Return Value

Numeric

Note

Returns a number from 1 to 31 for a valid date or 0 for an invalid date.

See Also

[CDOW\(\) Function](#)

[DOW\(\) Function](#)

[SET FDOW Command](#)

[SET FWEEK Command](#)

DBC()

DBC()

Remarks

Returns the name of the current database.

Example

```
* Interrogate the JAXCorp database that you installed  
* onto your MySQL or SQL Server database server using  
* the CreateDSNFile program in the TOOLS folder.
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
CLEAR
```

```
SET NAMING TO LOWER  
cConnect=DECRYPT(FILETOSTR(_JAXFolder+"\tools\dsn\JAXCorp.dsnx" ))  
OPEN DATABASE JAXCorp USING cConnect
```

```
? DBC()      && Returns JAXCORP  
CLOSE DATABASE  
? DBC()      && Return the empty string
```

Return Value

Character

Note

DBC() returns an empty string if there is no current database.

Use SET DATABASE to specify the current database.

See Also

[CREATE DATABASE](#)

[CLOSE DATABASE](#)

[OPEN DATABASE](#)

[SET DATABASE](#)

[SQL Class](#)

DBF()

`DBF([nWorkArea | cAlias])`

Remarks

Returns the name of a table open in a specified work area or opened with the specified alias.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Interrogate the JAXCorp database that you installed
* onto your MySQL or SQL Server database server using
* the CreateDSNFile program in the TOOLS folder.
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET NAMING TO LOWER
cConnect=DECRYPT(FILETOSTR(_JAXFolder+"\tools\dsn\JAXCorp.dsnx"))
OPEN DATABASE USING cConnect
SELECT 1
SELECT * FROM CUSTOMERS INTO CURSOR Cust

SELECT 15
USE (_JAXFolder+"data\counties")

? DBF("cust") && Returns jaxcorp!customers
? DBF(1)      && Returns counties
```

Return Value

Character

Note

If no parameter is given, the current work area is assumed.

If no table is open in the specified work area, an empty string is returned.

See Also

[FIELD\(\) Function](#)

[NDX\(\) Function](#)

[SET FULLPATH Command](#)

[USE Command](#)

DBGETPROP()

Status

Slated for Version 2.0

DRAFT

DBSETPROP()

Status

Slated for Version 2.0

DRAFT

DBUSED()

DBUSED(cDatabaseName)

Remarks

Returns a value of true (.T.) if the specified database is open.

Parameters

cDatabaseName

Character expression of the database to query.

Example

* Interrogate the JAXCorp database that you installed
* onto your MySQL or SQL Server database server using
* the CreateDSNFile program in the TOOLS folder.

```
CLEAR ALL  
CLOSE ALL  
CLEAR
```

```
SET NAMING TO LOWER  
cConnect=DECRYPT(FILETOSTR(_JAXFolder+"\tools\dsn\JAXCorp.dsdx" ))  
OPEN DATABASE USING cConnect  
  
? DBUSED( "JAXCORP" ) && Returns .T.
```

Return Value

Logical

Note

Returns the value true (.T.) if the database is open, otherwise returns false (.F.).

See Also

CLOSE DATABASE
CREATE DATABASE
OPEN DATABASE
SQL Class

DELETED()

DELETED([nWorkArea | cAlias])

Remarks

Returns whether the current record is deleted.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
GOTO TOP
DELETE
? DELETED()  && Returns .T.
RECALL
? DELETED()  && Returns .F.
```

Return Value

Logical

Note

Only local tables and cursors from tables or SQL tables can be queried with DELETED().

DELETED() returns a value of True (.T.) if the current record is marked for deletion.

See Also

[DELETE Command](#)

[PACK Command](#)

[RECALL Command](#)

[SET DELETED Command](#)

DESCENDING()

`DESCENDING([cIndexName | nIndexNumber [,nWorkArea | cAlias]])`

Remarks

Returns a value of true (.T.) if the indicated index or compound index tag is ordered in descending order.

Parameters

cIndexName

Character expression of an open index. 0 for the current selected index.

nIndexNumber

Numeric expression related to the list of open indexes.

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

Return Value

Logical

Note

DESCENDING() returns a value of true (.T) under two conditions.

The selected index was created with the DESCENDING keyword and opened normally.

The selected index with opened with the DESCENDING keyword in a USE, SET INDEX, or SET ORDER command.

If you do not include any parameters, then DESCENDING() looks at the controlling index in the current workarea.

If there is no index currently selected, DESCENDING() returns a value of false (.F.).

See Also

INDEX Command
SET INDEX Command
SET ORDER Command
USE Command

DRAFT

DIFFERENCE()

Status

Slated for Version 2.0

DRAFT

DIRECTORY()

DIRECTORY(cDirectoryName [,nFlags])

Remarks

Locates the specified directory.

Parameters

cDirectoryName

Character expression that specifies the name of the directory to check.

nFlags

Numeric expression where:

- 0 (default if omitted) returns false (.F.) if the directory exists but is Hidden or a System folder
- 1 causes DIRECTORY() to return true (.T.) if the directory exists regardless of its attributes

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET NAMING TO NATURAL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

IF "WINDOWS" $ OS
    ? DIRECTORY( "C:\WINDOWS" )    && Returns .T. in Windows
ELSE
    ? DIRECTORY( "/root" )        && Returns .T. in Linux
ENDIF
```

Return Value

Logical

Note

DIRECTORY() returns true (.T.) if the directory is found, otherwise false.

In Windows, if the directory does not have an absolute path, JAXBase searches the default directory for the provided directory expression.

Linux uses a unified filesystem tree where all drives and directories are listed under the root folder (/). If the folder in Linux does not start with the forward slash (/), then JAXBase searches the default directory for the provided directory expression.

See Also

ADIR() Function
CD Command
FILE() Function
GETDIR() Function
GETFILE() Function
MD Command
SET DEFAULT Command

DRAFT

DISKSPACE()

`DISKSPACE(cVolumeName [,nType])`

Remarks

Returns the number of bytes of the specified type that are available on the default or specified device or volume.

Parameters

cVolumeName

Character expression holding the specified device name or volume. If empty or omitted, returns the number of free bytes for the device or volume that holds the default directory.

nType

Numeric expression holding one of the following values which specify what to return:

Value	Description
1	Total bytes on the device or volume
2	Total free bytes on the device or volume (default)
3	Total amount of free space to the associated user

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
? DISKSPACE( "",1)
```

Return Value

Numeric

Note

DISKSPACE() number of bytes available. If an error occurs, it returns -1.

See Also

[HEADER\(\) Function](#)
[RECSIZE\(\) Function](#)
[SET DEFAULT Command](#)

DYM()

DYM(dExpression | tExpression)

Remarks

Returns a date in the dd-Month-year format.

Parameters

dExpression

Date expression to convert.

tExpression

DateTime expression to convert.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CENTURY ON
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? DYM(DATETIME())    && Returns current date in dd-Month-yyyy format
? DYM({09/06/2025}) && Returns 06-September-2025
```

Return Value

Character

Note

Returns the date portion of the expression in the dd-Month-yyyy format if CENTURY is set on.

Returns the date portion of the expression in the dd-Month-yy format if CENTURY is set OFF.

Returns an empty character value if an error occurs.

The month name is not abbreviated.

See Also

[DATE\(\) Function](#)

[DATETIME\(\) Function](#)

[MDY\(\) Function](#)

[SET CENTURY Command](#)

[SET DATE Command](#)

DODEFAULT()

DODEFAULT([eParameter1 [, eParameter2] ...])

Remarks

Executes the parent class event or method of the same name from within a subclass.

Parameters

eParameter1 [,eParameter2] ...

Specifies parameters that are passed to the parent class event or method.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

T=CREATECLASS( "TEST1" )
? T.ReturnProperty( "MYPROPERTY" )  && Displays 1
RETURN

DEFINE CLASS TEST0 AS COLLECTION
MyProperty=1

PROCEDURE ReturnProperty(cProp)
    RETURN EVALUATE("this."+cProp)
ENDPROCEDURE
ENDDDEFINE

DEFINE CLASS TEST1 AS TEST0
MyProperty=2

PROCEDURE ReturnProperty(cProp)
    RETURN DODEFAULT(cProp)
ENDPROCEDURE
ENDDDEFINE
```

Return Value

JAXBase supported data type or object

Note

DODEFAULT() returns the data type value that is sent from the called event or method.

If there is no parent class code to execute, DODEFAULT() returns true (.T.).

See Also

:: Scope Resolution Operator (Slated for Version 2)

ParentClass property

DOW()

```
DOW(dExpression | tExpression [, nFirstDayOfWeek])
```

Remarks

Returns a numeric day-of-the-week value from a Date or DateTime expression

Parameters

dExpression

Date expression used to get the DOW() value

tExpression

DateTime expression used to get the DOW() value

nFirstDayOfWeek

Accepted, but slated for Version 2. Currently, Sunday is the first day of the week.

Value	Description
0	Whatever day is set for first-day-of-the-week in the regional settings (default as of Ver 2)
1	Sunday (default prior to Version 2)
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? DOW(DATE())
? DOW(DATETIME())
```

Return Value

Numeric

See Also

[CDOW\(\) Function](#)
[DAY\(\) Function](#)
[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[SET FDOW Command](#)
[SET FWEEK Command](#)
[WEEK\(\) Function](#)

DRAFT

DRIVETYPE()

DRIVETYPE(cDevice)

Remarks

Returns the type of the specified device or volume.

Parameters

cDevice

In Windows, a device is a drive letter while in Linux, a device has a name that follows the convention sdX (example: sd0) or nvmeXnY (example: nvme0n1).

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
ADRIVES(aDriveList)
? DRIVETYPE(aDriveList[1])
```

Return Value

Numeric

Note

The values returned by DRIVETYPE() are:

Value	Description
1	No type or unknown
2	Floppy disk
3	Hard disk
4	Removable drive or network driver
5	Optical drive (CD, DVD, Blu-Ray)
6	RAM drive

See Also

[ADRIVES\(\) Function](#)

[JUSTDRIVE\(\) Function](#)

DTOC()

```
DTOC(dExpression | tExpression)
```

Remarks

Returns the Date or DateTime in the regional format.

Parameters

dExpression

Date expression used to return the value.

tExpression

DateTime expression used to return the value.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CENTURY ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET DATE TO AMERICAN
? DTOC(DATE())      && Returns date in mm-dd-yyyy format

SET DATE TO BRITISH
? DTOC(DATE())      && Returns the date in dd-mm-yyyy format
```

Return Value

Character

Note

The year will be displayed as yy if CENTURY is OFF.

If a DateTime value is provided, only the date portion will be displayed.

See Also

CTOD() Function

SET CENTURY Command

SET DATE Command

TTOC() Funciton

DTOR()

DTOR(nExpression)

Remarks

Transforms degrees to radians.

Parameters

nExpression

Numeric expression specifying degrees.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? DTOR(40)    && Returns 0.70
? DTOR(400)   && Returns 6.98
```

Return Value

Numeric

Note

The number of decimal points returned is determined by the SET DECIMALS command.

Use RTOD() to turn radians to degrees.

See Also

[ACOS\(\) Function](#)
[ASIN\(\) Function](#)
[ATAN\(\) Function](#)
[ATN2\(\) Function](#)
[COS\(\) Function](#)
[RTOD\(\) Function](#)
[SIN\(\) Function](#)
[TAN\(\) Function](#)

DTOS()

```
DTOS(dExpression | tExpression)
```

Remarks

Returns a date as a string of numbers in year month date format.

Parameters

dExpression

Date expression used to create the return value.

tExpression

DateTime expression used to create the return value.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? DTOS(DATE( ))      && Returns date portion
? DTOS(DATETIME( ))  && Returns date portion
```

Return Value

Character

Note

This function is useful for indexing Date or DateTime fields.

The format of the returned string is yyyyymmdd and is not affected by the SET CENTURY setting.

See Also

[DATE\(\) Function](#)

[DYM\(\) Function](#)

[DATETIME\(\) Function](#)

[CTOD\(\) Function](#)

[DTOC\(\) Function](#)

[DTOT\(\) Function](#)

DTOT()

`DTOT(dExpression | tExpression)`

Remarks

Turns a Date value into a DateTime value.

Parameters

dExpression

Date expression used to create the DateTime value.

tExpression

DateTime expression used to create the DateTime value.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
ltDate=DATE()  
? DTOT(ltDate)      && Returns DateTime with time portion set to midnight
```

Return Value

Character

Note

During conversion, the time portion is always set to midnight of the date.

The SET HOURS command determines how the time is displayed (24 for military, 12 for AM/PM) while the SET CENTURY and SET DATE command determines how the date portion is displayed.

See Also

[CTOD\(\) Function](#)

[DATE\(\) Function](#)

[DATETIME\(\) Function](#)

[SET DATE Command](#)

[SET HOURS Command](#)

[SET CENTURY Command](#)

[TIME\(\) Function](#)

EMPTY()

EMPTY(eExpression)

Remarks

Returns a logical value if eExpression evaluates to an empty value.

Parameters

eExpression

Expression to test.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? 0, EMPTY(0)
? 1, EMPTY(1)
? "", EMPTY(" ")
? "HI", EMPTY("HI")
? .F., EMPTY(.F.)
? .T., EMPTY(.T.)
```

Return Value

Logical

Note

All JAXBase data types have an empty value and EMPTY() will return true (.T.) if the expression evaluates to an empty value, as described in the following table.

Expression Type	Evaluates to
Binary or General field Varbinary data	0x0 or zero bytes
Character or Memo	Empty string
Currency	0
Date	Value equivalent to CTOD("")
DateTime	Value equivalent to CTOT("")
Double	0
Float	0
General	0x0 or zero bytes
Integer	0
Logical	.F.
Numeric	0

See Also

[LEN\(\) Function](#)
[TYPE\(\) Function](#)
[ISNULL\(\) Function](#)

DRAFT

EOF()

EOF([nWorkArea | cAlias])

Remarks

Returns true if the specified table is at the end of file.

Parameters

Example

```
* Demonstrate EOF()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
? EOF()

GOTO BOTTOM
? EOF()
SKIP
? EOF()

SKIP -1
? EOF()
```

Return Value

Logical

Note

EOF() returns true when the record pointer attempts to go past the last record of the file or indexed order. For instance, when FIND(), LOCATE(), SEEK or SEEK() are unsuccessful (and SET NEAR is OFF).

See Also

BOF()
GOTO Command
SET NEAR Command
SKIP Command

ERROR()

ERROR()

Remarks

Returns the last program error captured by JAXBase. An ON ERROR, TRY...CATCH, or Error method must be active to be able to use the ERROR() function.

Example

```
* Demonstrate ERROR()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

TRY
    ERROR 10
CATCH
    ? ERROR()
ENDTRY
```

Return Value

Numeric

Note

ERROR() returns the error number and MESSAGE() returns the error text related to the error.

ERROR() is reset by RETURN or RETRY.

See Also

ON ERROR Command
TRY/CATCH Command
ERROR Method

EVALUATE()

EVALUATE(cExpression)

Remarks

Evaluates a character expression.

Parameters

cExpression

Character expression representing an eExpression.

Example

```
* Demonstrate EVALUATE()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
A="2+3"
```

```
? EVALUATE(A) && Returns 5
```

```
A="2>3"
```

```
? EVALUATE(A) && Returns .F.
```

```
B=7
```

```
A="B+2"
```

```
? EVALUATE(A) && Returns 9
```

Return Value

JAXBase data type or object

Note

cExpression must represent a valid JAXBase expression or an error will be generated.

Maximum cExpression length is 255 characters.

EVALUATE() is usually faster than macro substitution, so use it whenever possible.

See Also

[TYPE\(\) Function](#)

[VARTYPE\(\) Function](#)

EVL()

EVL(eExpression1, eExpression2)

Remarks

Returns a non-empty value of two expressions.

Parameters

eExpression1

JAXBase expression to evaluate.

eExpression2

JAXBase expression to return if eExpression1 evaluates to an empty value.

Example

```
* Demonstrate EVL()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? EVL("",1)    && Returns 1
? EVL(1,7)    && Returns 1
```

Return Value

JAXBase data type or object

Note

eExpression2 does not have to be of the same data type as eExpression1.

If eExpression1 evaluates to an empty value, then eExpression2 is returned.

See Also

[EMPTY\(\) Function](#)

EXECSCRIPT()

```
EXECSCRIPT(cExpression [,eParameter1 ,eParameter2, ...])
```

Remarks

Enables you to run a character expression holding multiple lines of JAXBase code.

Parameters

cExpression

Character expression holding one or more lines of JAXBase source code.

Example

```
* Demonstrate EXECSCRIPT()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

aString=? [HELLO]"+chr(13)+"? [GOODBYE!]"
EXECSCRIPT(aString)
```

Return Value

If the script has a RETURN statement, the value passed back will be returned, otherwise a value of true (.T.) is returned.

Note

Code executed by EXECSCRIPT() runs in its own program level, so local variables in the program with EXECSCRIPT are not accessible by the script being run.

If you pass parameters using EXECSCRIPT() then the first line of executable code must be a parameter statement.

See Also

APARAMETERS Command

LPARAMETERS Command

PARAMETERS Command

ON ERROR Command

RETURN Command

Macro Substitution

EXP()

EXP(nExpression)

Remarks

Returns the value of e^x where x is a specified numeric expression.

Parameters

nExpression

Numeric expression specifying the exponent x , in the exponential expression e^x .

Example

```
* Demonstrate EXP( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? EXP(0)      && Displays 1.00
? EXP(1)      && Displays 2.72
? EXP(2)      && Displays 7.39
```

Return Value

Numeric

See Also

[LOG\(\) Function](#)

[LOG10\(\) Function](#)

FCOUNT()

FCOUNT(nWorkArea | cAlias)

Remarks

Returns the number of fields in the specified table or cursor.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate FCOUNT()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
USE (_JAXFolder+"data\counties")
? FCOUNT()
```

Return Value

Numeric

Note

If you do not give it parameters, FCOUNT() uses the current work area. If a work area number was specified, or no parameter given, and a table is not open then FCOUNT() will return 0. If an alias is provided and does not exist, FCOUNT() will generate an error.

See Also

[DBF\(\) Function](#)

[FIELD\(\) Function](#)

[FSIZE\(\) Function](#)

FDATE()

FDATE(cFileName)

Remarks

Returns the last modified DateTime of the specified file.

Parameters

cFileName

A character expression indicating the name of a file.

Example

```
* Demonstrate FDATE()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? FDATE(_JAXFolder+"data\counties.dbf")
```

Return Value

Numeric

Note

FDATE() returns the last modified date of a file. If a Fully Qualified File Name (FQFN) is not given, then JAXBase will search for the file in the default path list. If the file is not found then an error is generated.

See Also

DATETIME() Function
FTIME() Function
LUPDATE() Function
FULLPATH() Function
SET PATH Command

FIELD()

```
FIELD(nFieldNumber [,WorkArea | cAlias])
```

Remarks

Returns the specified field name or caption.

Parameters

nFieldNumber

A numeric expression indicating which field name from the field list to return.

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate FIELD()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
USE (_JAXFolder+"data\counties")
? FIELD(1)
? FIELD(2)
```

Return Value

Character

Note

If there is no table open in the work area, an error is generated.

See Also

DISPLAY STRUCTURE Command

FCOUNT() Function

FSIZE() Function

FILE()

FILE(cFileName [,nFlags])

Remarks

Returns a value of true (.T.) if the specified file exists.

Parameters

cFileName

Character expression of the file name to search.

nFlags

Numeric expression where 0 indicates that Hidden and System files should be ignored, while 1 indicates that all files are checked.

Example

```
* Demonstrate FILE()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? FILE(_JAXFolder+"data\counties.dbf")
```

Return Value

Logical

Note

If the cFileName is not a Fully Qualified File Name (FQFN), then JAXBase will search for the file in the path list.

Returns a value of true (.T.) if the file is found.

See Also

[ADIR\(\) Function](#)

[CD Command](#)

[FULLPATH\(\) Function](#)

[GETFILE\(\) Function](#)

[LOCFILE\(\) Function](#)

FILETOSTR()

FILETOSTR(cFileName)

Remarks

Returns the contents of the specified file.

Parameters

cFileName

Character expression of the file name to search.

Example

```
* Demonstrate FILETOSTR  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? FILETOSTR(_JAXFolder+"data\states.dbf")
```

Return Value

Character

Note

If the cFileName is not a Fully Qualified File Name (FQFN), then JAXBase will search for the file in the path list.

The contents of the file will be returned as a string.

If the file does not exist, an error will be generated.

See Also

[STRTOFILE\(\) Function](#)

[JAXBase System Capacities](#)

FILTER()

`FILTER(nWorkArea | cAlias)`

Remarks

Returns the current filter to the specified work area or alias.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate FILTER()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties.dbf")
SET FILTER TO State="WI"
? FILTER()
COUNT TO A
? RECCOUNT(), A
```

Return Value

Character

Note

If an alias is given and it does not exist, an error is generated. If a work area is given or the parameter omitted, and there is no open table, then an empty character string is returned.

See Also

[SET FILTER Command](#)

[Filter Property](#)

FLOCK()

FLOCK(nWorkArea | cAlias)

Remarks

Attempts to lock the specified DBF table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate FLOCK()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties.dbf")

IF FLOCK()
    ? "FILE IS LOCKED"
ELSE
    ? "FAILED TO LOCK"
ENDIF

CLOSE ALL
```

Return Value

Logical

Note

When a table is locked, the table is available to the program that placed the lock. Other users on the network have read-only access to the table. To lock a table and prevent access by other users, see SET EXCLUSIVE Command and USE Command.

A table remains locked until it is unlocked. The table can be unlocked by UNLOCK, closing the table, or quitting JAXBase. Tables can be closed with USE, CLEAR ALL, or CLOSE DATABASES.

By default, FLOCK() attempts to lock a table once. Use SET REPROCESS to automatically retry a table lock when the first attempt fails. SET REPROCESS determines the number of lock attempts, or the length of time during which lock attempts are made when the initial lock attempt is unsuccessful.

A table is locked only when an explicit FLOCK() is placed against it.

Care must be taken when attempting to lock two or more tables as it is possible that someone on the network may be attempting to place locks on multiple tables at the same time and a death lock occurs. For an example of death locks and how to avoid them, look at the `deathlock.prg` program in `SAMPLES`.

See Also

`CLEAR` Commands
`CLOSE DATABASES` Command
`CLOSE TABLES` Command
`ISFLOCKED()` Function
`ISRLOCKED()` Function
`LOCK()` Function
`RLOCK()` Function
`UNLOCK` Command
`USE`

FLOOR()

FLOOR(nExpression)

Remarks

Returns the next lowest integer that is less than or equal to the specified value.

Parameters

nExpression

The numeric expression used to find the FLOOR() value.

Example

```
* Demonstrate FLOOR()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? FLOOR(100.3)      && Returns 100
? FLOOR(-50.7)      && Returns -51
? FLOOR(10)          && Returns 10
```

Return Value

Numeric

See Also

[CEILING\(\)](#)
[INT\(\)](#)
[ROUND\(\)](#)

FONTMETRIC()

Status

Slated for Version 2

DRAFT

FOR()

```
FOR(nIndexNumber [, nWorkArea | cAlias])
```

Remarks

Returns the FOR filter expression for the specified index.

Parameters

nIndexNumber

Numeric expression indicating which index to interrogate from the index list.

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate FOR()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties.dbf")
SET INDEX TO COUNTIES_STATE
SET INDEX TO COUNTIES_NAME

? FOR(2)
```

Return Value

Charcter

Note

If the second parameter is not provided, the current work area is use.

If an alias is given and not found, an error is generated.

If there is no index open or the nIndexNumber goes beyond the limits of the index list, and empty character value is returned.

See Also

INDEX Command

FORCEEXT()

FORCEEXT(cFileName, cExtension)

Remarks

Returns a string with the specified filename with its old extension replaced by the specified extension.

Parameters

cFileName

String containing a file name.

cExtension

String containing a file extension without the leading period.

Example

```
* Demonstrate FORCEEXT()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET NAMING TO LOWER
? FORCEEXT( "counties.dbf" , "BAK" ) && Returns counties.bak
? FORCEEXT( "counties" , "BAK" ) && Returns counties.bak
? FORCEEXT( "counties.dbf." , "BAK" ) && Returns counties.dbf.bak
```

Return Value

Character

Note

What FORCEEXT() returns is dictated by the SET NAMING command.

FORCEEXT() does not do any checking. It assumes that it is being given a legal PathFile name.

See Also

ADDBS() Function

DEFALTEXT() Function

FILE() Function

FORCEPATH() Function

JUSTDRIVE() Function

JUSTTEXT() Function

JUSTFNAME() Function

JUSTFULLPATH() Function

JUSTPATH() Function

JUSTSTEM() Function

SET NAMING Command

FORCEPATH()

FORCEPATH(cFileName, cPath)

Remarks

Returns a string where the original file path is replaced with the specified file path.

Parameters

cFileName

String containing a file name.

cPath

String containing the new path.

Example

```
* Demonstrate FORCEPATH()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
SET NAMING TO LOWER
? FORCEPATH( "counties.dbf" , "C:\DATA" )           && Returns c:\data\counties.dbf
? FORCEPATH( "" , "C:\DATA" )                         && Returns c:\data\
? FORCEPATH( "C:\TEST.TXT" , "C:\DATA" )             && Returns c:\data\counties.dbf
? FORCEPATH( "C:\TEST" , "C:\DATA" )                  && Returns c:\data\test
? FORCEPATH( "C:\TEST\" , "C:\DATA" )                 && Returns c:\data\
```

Return Value

Character

Note

What FORCEPATH() returns is dictated by the SET NAMING command. Further, FORCEPATH() does not do any checking. It assumes that it is being given a legal Path and File information.

See Also

[ADDBS\(\) Function](#)

[DEFAUTTEXT\(\) Function](#)

[FILE\(\) Function](#)

[FORCEPATH\(\) Function](#)

[JUSTDRIVE\(\) Function](#)

[JUSTTEXT\(\) Function](#)

[JUSTFNAME\(\) Function](#)

[JUSTFULLPATH\(\) Function](#)

[JUSTPATH\(\) Function](#)

[JUSTSTEM\(\) Function](#)

[SET NAMING Command](#)

FOUND()

FOUND([nWorkArea, cAlias])

Remarks

Returns a value of true (.T.) if the last LOCATE command, SEEK command, or SEEK() function successfully found an exact matching record.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate FOUND()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
USE (_JAXFolder+"data\counties")
SET INDEX TO COUNTIES_BYSTATE
LOCATE FOR STATE="FL"
? FOUND()
SEEK "CAVENTURA"
? FOUND()
```

Return Value

Logical

Note

Returns a value of true (.T.) if the most recent FIND, LOCATE, SEEK or CONTINUE command was successful. FOUND() is also affected by the SEEK() function.

If there is no table open in the work area you specify, FOUND() returns false (.F.). If you specify an alias and it does not exist, an error is generated.

See Also

CONTINUE Command
FIND Command
LOCATE Command
SEEK Command
SEEK() Function
EOF() Function

FSIZE()

```
FSIZE(cFieldName [, nWorkArea | cAlias])
FSIZE(cFilename)
```

Remarks

Returns the size in bytes of the specified file.

Parameters

cFieldName

Character expression containing a field name.

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

cFileName

Character expression containing a file name.

Example

```
* Demonstrate FSIZE()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? "File size ",FSIZE(_JAXFolder+"data\counties.dbf")
USE (_JAXFolder+"data\counties")
? "Field size",FSIZE("name")
```

Return Value

Numeric

Note

If the cFileName is not a Fully Qualified File Name (FQFN), then JAXBase will search for the file in the path list. If found, it will return the size in bytes. If not, it will return 0.

For field size, if there is no table open in the work area you specify, FSIZE() returns 0. If you specify an alias and it does not exist, an error is generated.

When a field size is returned, it should be noted that the following fields always return the following values.

Field Type	Size Returned	
Currency	8	Currency, DateTime, Double, and Integer fields are binary encoded character fields.
Date	8	Dates are stored as text data in yyyyymmdd format.
DateTime	8	
Double	8	
Integer	4	Memo, Blob, and General fields are integer values that are pointers to the location of the data in the FPT file.
Logical	1	
Memo	4	
Blob	4	Logical fields hold the characters T or F which is translated to .T. and .F. values.
General	4	

Field sizes can also be obtained using the DISPLAY STRUCTURE command or AFIELDS() function.

See Also

- [AFIELDS\(\) Function](#)
- [DISPLAY STRUCTURE Command](#)
- [FCOUNT\(\) Function](#)
- [Table File Structure](#)
- [JAXBase Data Types](#)
- [DBF Files](#)

FTIME()

FTIME(cFileName)

Remarks

Returns the last modified date of the specified file.

Parameters

cFileName

Character expression holding the file name to get the last modified date.

Example

```
* Demonstrate FTIME()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? FTIME(_JAXFolder+"data\counties.dbf")
```

Return Value

Character

Note

If the cFileName is not a Fully Qualified File Name (FQFN), then JAXBase will search for the file in the path list. If found, it will return the last modified time. If not, it will return an empty character string.

See Also

FDATE() Function
SET PATH Command
TOSEC() Function
TOTIME() Function

FULLPATH()

FULLPATH(cFileName)

Remarks

Returns a string representing a file with an absolute path.

Parameters

cFileName

Character expression holding a file or partial path and file name.

Example

```
* Demonstrate FULLPATH() assuming that
* the C:\TEMP folder exists
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
SET NAMING TO LOWER
ACTIVATE CONSOLE
CLEAR

SET DEFAULT TO C:\TEMP
=STRTOFILE("test", "TEST.txt")
? FULLPATH("test.txt")           && Returns c:\temp\test.txt
```

Return Value

Character

Note

FULLPATH() returns the file name with the absolute path, making it easier to address a file or pass a file with path to another module or program.

FULLPATH() is affected by the SET NAMING command.

If cFileName is not found, FULLPATH() will search the path list, and if found, return the file with an absolute path. If it is not found, an empty character string is returned.

See Also

[DBF\(\) Function](#)

[FILE\(\) Function](#)

[GETFILE\(\) Function](#)

[LOCFILE\(\) Function](#)

[SET NAMING Command](#)

FV()

```
FV(nPayment, nInterestRate, nPeriods)
```

Remarks

Returns the present value amount of each payment (positive or negative).

Parameters

nPayment

Numeric expression specifying the periodic payment amount.

nInterestRate

Numeric expression specifying the interest rate per period. If calculating monthly payments with annual interest, divide the interest rate by 12.

nPeriods

Numeric expression specifying the total number of payments to be made.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
? FV(500, .075/12, 48)      && Displays 27887.93
```

Return Value

Numeric

Note

FV() assumes on-time payments earning compound interest providing the ability to create an investment plan.

See Also

CALCULATE Command

FV() Function

PV() Function

GETAUTOINCVALUE()

Status

Slated for Version 2.0

DRAFT

GETCP()

Status

Slated for Version 2.0

DRAFT

GETDATE()

GETDATE([lDateOnly, [dExpression | tExpression]])

Remarks

Opens the Date/Time dialog and return a Dates or DateTime value based lDateOnly setting.

Parameters

lDateOnly

Logical expression, if true (.T.) limits entry to a Date value, otherwise accepts Date and Time entry.

dExpression

Date expression for the starting date.

tExpression

DateTime expression for the starting date and time.

Example

```
* Demonstrate GETDATE()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? GETDATE()
```

Return Value

Date value if lDateOnly is true (.T.) and DateTime value if lDateOnly is false (.F.) or omitted.

Note

If lDateOnly is true (.T.), the time controls are greyed out. Default is false (.F.) allowing the user to also set a time component.

If dExpression is provided and lDateOnly is false (.F.), the time component defaults to midnight.

If no Date or DateTime expression is provided, the current OS time is assumed. The time component is affected by SET HOURS command.

See Also

[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[GETTIME\(\) Function](#)

GETDIR()

```
GETDIR([cDirectory [, cText [, cCaption [, nFlags, [lRootOnly]]]]])
```

Remarks

Displays the Select Directory dialog box and returns the absolute directory path selected.

Parameters

cDirectory

A character expression indicating the starting point.

cText

Character expression to display under the caption.

cCaption

Character expression displayed in the caption.

nFlags

Numeric expression where the following values can be added together as needed to alter how the Select Directory dialog works. The default is to not display hidden and system folders, not include the edit box, not validate a path, and not create a path if it does not exist.

nFlag	Description
1	Display hidden and system folders
2	Include the edit box allowing the user to type in a directory path
4	Validate the path selected or entered
8	Create the path entered by the user if it does not exist

lRootOnly

If a value of true (.T.) then the user cannot navigate to another drive nor to a directory above the current.

Example

```
* Demonstrate GETDIR()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

* Include the edit box and validate the directory path
? GETDIR(_JAXFolder, "Select", "Please select a directory", 2+4)
```

Return Value

Character

Note

If a directory is not selected or the user cancels the selection, an empty character string is returned.

Any error generated by the Select Directory dialog will produce an error dialog informing the user of the issue, but no JAXBase error is generated and the user will be allowed to make changes and try again.

See Also

[DIR or DIRECTORY Command](#)

[DIRECTORY\(\) Function](#)

[GETFILE\(\) Command](#)

DRAFT

GETENV()

GETENV(cVariableName)

Remarks

Returns the specified operating system environment variable as a string.

Parameters

cVariableName

Character expression specifying the environment variable's value to return.

Example

```
* Demonstrate GETENV()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

* JAXBase converts common environment
* variables between Windows and Linux.
? GETENV("$HOME")
? GETENV("%USERPROFILE%")
```

Return Value

Character

Note

JAXBase converts common Windows and Linux environment variables.

Windows	Linux	Description
%USERPROFILE%	\$HOME	User's home path
%TEMP%	\$TEMP or /tmp	Operating system's temporary folder
PATH	PATH	Operating path list

See Also

[DISKSPACE\(\) Function](#)

[OS\(\) Function](#)

[VERSION\(\) Function](#)

GETFILE()

```
GETFILE([cFileExtensions] [,cText [,cOpenButtonCaption [,nButtonType [,cCaption]]]])
```

Remarks

Displays the Open File dialog box.

Parameters

cFileExtensions

Character expression with list of comma or semicolon delimited extensions.

cText

Character expression to display in the File Name label.

cOpenButtonCaption

Character expression to display in the Open Button's caption.

nButtonType

Value	Buttons Displayed
0 or omitted	OK, Cancel
1	OK, New, Cancel
2	OK, None, Cancel

cCaption

Character expression to display as the dialog caption.

Example

```
* Demonstrate GETFILE()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? GETFILE("dbf, idx", "Select", 2, "Please select a file")
```

Return Value

Character

Note

GETFILE() returns the string Untitled if the New or None buttons are Clicked and returns an empty character string if the Cancel button is clicked.

See Also

[FULLPATH\(\) Function](#)
[GETDIR\(\) Function](#)
[GETPICT\(\) Function](#)
[LOCFILE\(\) Function](#)
[PUTFILE\(\) Function](#)

DRAFT

GETFIELDSTATE()

Status

Slated for Version 2.0

DRAFT

GETFONT()

Status

Slated for Version 2.0

DRAFT

GETNEXTMODIFIED()

Status

Slated for Version 2.0

DRAFT

GETJSON() Function

GETJSON(cJSONString,cFieldToReturn)

Remarks

Returns a field value from a JSON string.

Parameters

cJSONString

Character expression holding the JSON string to parse.

cFieldToReturn

Character expression holding the name of the JSON field to return.

Example

```
* DEMONSTRATE GETJSON( )
SET TALK OFF
CLEAR
JSONString=[{"Name": "Fred", "BDate": 1979/07/05, "Balance": 140.73, "LastIn": "2025-04-17 13:41:00"}]
Name=JSON(JSONString, "Name")
BirthDate=JSON(JSONString, "Bdate")
Balance=JSON(JSONString, "balance")
LastIn=JSON(JSONString, "lastin")
? VARTYPE(Name), Name
? VARTYPE(BirthDate), BirthDate
? VARTYPE(Balance), Balance
? VARTYPE(LastIn), LastIn
```

Return Value

Supported JAXBase data type

Note

The JSON will parse a JSON string and return the value of the JASON field from the JSON string. The data type returned depends on the JSON string, but will one of the following

JSON Conversion	
A	Array containing elements from the JSON field
C	Character
D	Date
L	Logical
N	Numeric (including integer, float, double, etc)
O	Returns an EMPTY object with fields/values of the JSON object
T	DateTime
U	Unknown or invalid field name
X	.NULL.

A JSON array will change the receiving variable into a JAXBase array variable, while a JSON object will change the receiving variable into a JAXBase object variable using the EMPTY class.

See Also

EMPTY Class
PUTJSON() Function
JAXBase Data Types

DRAFT

GETPICT()

GETPICT([cFileExtensions][, cFileNameCaption[,cOpenButtonCaption]])

Remarks

Displays the Open Picture dialog box and returns the name of the picture chosen.

Parameters

cFileExtensions

Character expression with list of comma or semicolon delimited extensions.

cFileNameCaption

Character expression to display in the File Name label.

cOpenButtonCaption

Character expression to display in the Open Button's caption.

Example

```
* Demonstrate GETPICT()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET DEFAULT TO (_JAXFolder+"GRAPHICS")
? GETPICT("BMP,JPG,PNG","Select","Select")
```

Return Value

Character

Note

Returns the absolute file path of the selected picture unless the CANCEL button is clicked or the dialog is exited with the close control, then an empty character string is returned.

cFileExtensions simply filters the directory listing, but can be overridden by the user if they enter a file skeleton in the edit box and hit the ENTER key.

No file type validation is performed by GETPICT().

See Also

[GETFILE\(\) Function](#)

[LOCFILE\(\) Function](#)

[PUTFILE\(\) Function](#)

GETPRINTER()

GETPRINTER()

Remarks

Allows the user to select a printer from the available printers and returns a PRINTER class object for the selected printer.

Example

```
* Demonstrate GETPRINTER( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

oPrt=GETPRINTER( )
IF VARTYPE(oPrt)="O"
    ? oPrt.Name
ELSE
    ? "User did not select a printer"
ENDIF
```

Return Value

JAXBase Printer object

Note

If a printer is not selected, a .NULL. is returned.

See Also

APRINTERS() Function
SET PRINTER Command

GETTIME()

GETTIME(cTime | nSeconds)

Remarks

Opens the Time dialog and returns a character Time value.

Parameters

cTime

Character expression in a valid 12 or 24 hour format.

nSeconds

A numeric expression from 0 to 86399 representing 00:00:00 to 23:59:59.

Example

```
* Demonstrate GETTIME()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
SET HOURS TO 12
```

```
? GETTIME()
```

```
SET HOURS TO 24
```

```
? GETTIME(TIME())
```

Return Value

Character

Note

If no Time expression is provided, the current OS time is assumed.

The time format is affected by SET HOURS command. The value returned will be in either the 12 or 24 hour format, based on the SET HOURS setting.

See Also

[DATE\(\) Function](#)

[DATETIME\(\) Function](#)

[GETTIME\(\) Function](#)

[SET DATE Command](#)

[SET HOURS command](#)

[TTOSEC\(\)](#)

GETWORDCOUNT()

`GETWORDCOUNT(cExpression [,cDelimiters])`

Remarks

Counts the words in a string

Parameters

cExpression

Specifies the character expression whose words will be counted.

cDelimiters

The character expression that contains one or more delimiting characters.

Example

```
* Demonstrate GETWORDCOUNT()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

lcString="Fourscore and seven years ago our fathers brought forth,"+CHR(13)
      +"on this continent, a new nation, conceived in liberty, and"+CHR(13)
      +"dedicated to the proposition that all men are created equal."+CHR(13)
      +"Now we are engaged in a great civil war, testing whether that"+CHR(13)
      +"nation, or any nation so conceived, and so dedicated, can "+CHR(13)
      +"long endure."

? GETWORDCOUNT(lcString)    && Returns 53
```

Return Value

Numeric

Note

The default delimiting characters are space, tab, carriage return, and line feed.

GETWORDCOUNT uses each character in cDelimiters separately to get the word count.

If one or more delimiting characters are grouped together, that group will be counted as one delimiter.

See Also

[GETWORDNUM\(\)](#)

GETWORDNUM()

GETWORDNUM(cExpression, nIndex, cDelimiters)

Remarks

Returns the indicated word from a string of words.

Parameters

cExpression

Character expression containing the string to search.

nIndex

Numeric expression indicating which word to return'

cDelimiters

Character string containing the individual characters use as delimiters.

Example

```
* Demonstrate GETWORDNUM( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

lcString="Fourscore and seven years ago our fathers brought forth,"+CHR(13)
+"on this continent, a new nation, conceived in liberty, and"+CHR(13)
+"dedicated to the proposition that all men are created equal."+CHR(13)
+"Now we are engaged in a great civil war, testing whether that"+CHR(13)
+"nation, or any nation so conceived, and so dedicated, can "+CHR(13)
+"long endure."

? GETWORDNUM(lcString,22)           && Returns "the"
```

Return Value

Character

Note

The default delimiting characters are space, tab, carriage return, and line feed.

GETWORDNUM uses each character in cDelimiters separately to get the word count.

If one or more delimiting characters are grouped together, that group will be counted as one delimiter.

See Also

GETWORDCOUNT()

GETCURSORADAPTER()

Status

Slated for Version 2.0



GOMONTH()

`GOMONTH(dExpression | tExpression, nNumberOfMonths)`

Remarks

Returns the date that is a specified number of months before or after a given Date or DateTime expression.

Parameters

`dExpression`

Starting Date expression.

`tExpression`

Starting DateTime expression.

`nNumberOfMonths`

Numeric expression, negative or positive, indicating how many months before or after the specified Date or DateTime.

Example

```
* Demonstrate GOMONTH()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

FOR I=-1 TO -10 STEP -1
    ? GOMONTH(DATE(),I)
ENDFOR
```

Return Value

Date

Note

GOMONTH() does not support dates earlier than 1752.

See Also

[CMONTH\(\) Function](#)

[DATETIME\(\) Function](#)

[GETDATE\(\) Function](#)

[GETDATETIME\(\) Function](#)

[GETTIME\(\) Function](#)

GUID()

GUID()

Remarks

Returns a case sensitive 36-character GUID string in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

Example

```
* Demonstrate GUID()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? GUID()
? GUID()
? GUID()
? GUID()
```

Return Value

Character

HEADER()

HEADER([nWorkArea | cAlias])

Remarks

Returns the number of bytes in the header of the specified table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate HEADER()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
USE (JAXFolder+"data\counties.dbf")
? HEADER()
```

Return Value

Numeric

Note

If no parameter is given, the current work area is assumed. If an alias is provided and it is not found, an error is generated.

See Also

[FSIZE\(\) Function](#)

[RECSIZE\(\) Function](#)

HEXToint()

HEXToint(cHex)

Remarks

Returns a numeric value based on the provided character expression.

Parameters

cHex

Character expression containing a valid hexadecimal expression.

Example

```
* Demonstrate HEXToint()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? HEXToint( "F" )
? HEXToint( "0x01FA" )
```

Return Value

Numeric value

Note

The hex string may start with "0x" or "0X", which will be trimmed.

Hex strings start at 0x0 with a maximum value of 0xFFFFFFFFFFFFFF.

See Also

[INTTOHEX\(\)](#)

HOUR()

HOUR(dExpression | tExpression)

Remarks

Returns the hour from a DateTime expression.

Parameters

dExpression

Expression holding a Date value.

tExpression

Expression holding a DateTime value.

Example

```
* Demonstrate HOUR()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? HOUR(DATETIME())
```

Return Value

Numeric

Note

If dExpression is provided, HOUR() always returns 0.

The return value is always based on the 24 hour clock.

See Also

DATE()
DATETIME()
GETDATE()
GETTIME()
TIME()

ICASE()

```
ICASE(lCondition1, eResult1 [, lCondition2, eResult2]..., eOtherwiseResult)
```

Remarks

Evaluates the result from a list of conditions.

Parameters

lCondition

Expression that evaluates to a logical true (.T.) or false (.F.)

eResult

An expression that is a valid JAXBase data type or object.

eOtherwiseResult

An expression that is a valid JAXBase data type or object.

Example

```
* Demonstrate ICASE()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
? "Good "+ICASE(HOURS(SECONDS())<12, "morning!", HOURS(SECONDS())<18, "afternoon!", "Evening!")
```

Return Value

A valid JAXBase data type or object.

Note

ICASE() returns the eResult after the first lCondition that evaluates to true (.T.) or the eOtherwiseResult.

You can pass up to 100 pairs of parameters for ICASE().

See Also

IIF() Function

EVALUATE() Function

EVL() Function

IDXCOLLATE()

Status

Slated for Version 2.0

DRAFT

IIF()

`IIF(lCondition, eExpression1, eExpression2)`

Remarks

Returns one of two expressions based on the provided condition.

Parameters

lCondition

Logical condition expression.

eExpression1

Expression to return if lCondition is true (.T.)

eExpression2

Expression to return if lCondition is false (.F.)

Example

```
* Demonstrate IIF()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

IIF(HOUR(SECONDS())<12, "AM", "PM")
```

Return Value

JAXBase data type or object

Note

Also known as an immediate IF, can replace simple IF / ENDIF statements and is especially useful in report and label expressions to conditionally specify field contents.

See Also

[ICASE\(\) Function](#)

[IF / ENDIF Command](#)

[IF Statements](#)

INDEXSEEK()

```
INDEXSEEK(eExpression [, nWorkArea | cAlias [, nIndex | cIndexName]])
```

Remarks

Performs a SEEK() function without moving the record pointer and returns the record number.

Parameters

eExpression

Expression that holds the value to search for in the index.

nWorkArea

Numeric expression indicating the work area to check. 0 indicates the current work area.

cTableAlias

Character expression indicating the alias of the table to check.

nIndex

Index number from the open index list.

cIndexName

Index name from the open index list.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
GOTO TOP
? RECNO(),state,county
? INDEXSEEK("YORK",0,"counties_name")
? RECNO(),state,county
```

Return Value

Numeric

Notes

If a match is not found, -1 is returned.

INKEY()

`INKEY([nSeconds] [,cHideCursor])`

Remarks

Returns a numeric value corresponding to the first key press or mouse.

Parameters

nSeconds

Specifies the number of seconds to wait for a keystroke.

cHideCursor

Shows or hides the cursor or performs other tasks based on the following.

cHideCursor	Description
S	Show cursor (default)
H	Hide cursor
M	Check for a mouse click
E	Expand the keyboard macro if assigned. Read the entire macro using successive INKEYS with cHideCursor containing E.

Example

* Demonstrate INKEY()

CLEAR ALL

CLOSE ALL

SET EXCLUSIVE ON

SET CONSOLE ON

ACTIVATE CONSOLE

CLEAR

? "Press a key"

? "Key pressed = ", INKEY(0)

Return Value

Numeric

Note

If you do not specify a value for nSeconds, INKEY() immediately returns a value for a keystroke. If nSeconds is 0, INKEY() waits indefinitely for a keystroke.

INKEY Values

KEY	Alone	SHIFT	CTRL	ALT
F1	28	84	94	104
F2	-1	85	95	105
F3	-2	86	96	106
F4	-3	87	97	107
F5	-4	88	98	108
F6	-5	89	99	109
F7	-6	90	100	110
F8	-7	91	101	111
F9	-8	92	102	112
F10	-9	93	103	113
F11	133	135	137	139
F12	134	136	138	140
0	48	41		19
1	49	33		120
2	50	64		121
3	51	35		122
4	52	36		123
5	53	37		124
6	54	94		125
7	55	38		126
8	56	42		127
9	57	40		128
a	97	65	1	30
b	98	66	2	48
c	99	67	3	46
d	100	68	4	32
e	101	69	5	18
f	102	70	6	33
g	103	71	7	34
h	104	72	127	35
i	105	73	9	23
j	106	74	10	36
k	107	75	11	37
l	108	76	12	38
m	109	77	13	50
n	110	78	14	49
o	111	79	15	24

KEY	Alone	SHIFT	CTRL	ALT
p	112	80	16	25
q	113	81	17	16
r	114	82	18	19
s	115	83	19	31
t	116	84	20	20
u	117	85	21	22
v	118	86	22	47
w	119	87	23	17
x	120	88	24	45
y	121	89	25	21
z	122	90	26	44
INS	22	22	146	162
HOME	1	55	29	151
DEL	7	7	147	163
END	6	49	23	159
PAGE UP	18	57	31	153
PAGE DOWN	3	51	30	161
UP ARROW	5	56	141	152
DONN ARROW	24	50	145	160
RIGHT ARROW	4	54	2	157
LEFT ARROW	19	52	26	155
ESC	27	27	27	1
ENTER	13	13	10	166
BACKSPACE	127	127	127	14
TAB	9	15	148	
SPACEBAR	32	32	32	57
` or ~	96	126		41
- or _	45	95		
+ or =	61	43		13
[or {	91	123	27	26
] or }	93	125	29	27
\ or	92	124	28	43
; or :	59	58		39
' or "	39	34		40
, or <	44	60		51
. or >	46	62		52
/ or ?	47	63		53

See Also

[KEYBOARD Command](#)
[KeyPress Event](#)
[LASTKEY\(\)](#)
[ON KEY LABEL Command](#)
[SET TYPEAHEAD Command](#)

INLIST()

```
INLIST(eExpression1, eExpression2 [, eExpression3 ...])
```

Remarks

Indicates if an expression matches an of a set of expressions.

Parameters

eExpression1

Expression to test.

eExpression2 [,eExpresion3...]

List of expressions to test against.

Example

```
* Demonstrate INLIST()
```

```
CLEAR ALL
```

```
CLOSE ALL
```

```
SET EXCLUSIVE ON
```

```
SET CONSOLE ON
```

```
ACTIVATE CONSOLE
```

```
CLEAR
```

```
? INLIST("C", "A", "B", "C", "D", "E")
```

```
? INLIST("c", "A", "B", "C", "D", "E")
```

Return Value

Logical

Note

Returns true (.T.) if the first expression matches one of included list elements.

All expressions must be of the same type.

INLIST() is case sensitive.

See Also

\$ Operator

BETWEEN() Function

AT() Function

ATLINE() Function

OCCURS() Function

RAT() Function

RATLINE() Function

INPUTBOX()

```
INPUTBOX(cInputPrompt [,cCaption [,cDefaultValue [,nTimeout [,cTimeoutValue [,cCancelValue]]]])
```

Remarks

Displays the modal dialog for input of a single string.

Parameters

cInputPrompt

Character expression to place above the input box as a prompt with 64 characters being the maximum length allowed.

cDialogCaption

Character expression to place in the caption of the dialog.

cDefaultValue

Character expression put into the input box.

nTimeout

Numeric expression given in milliseconds on how long to wait after the most recent keystroke before exiting the input box.

cTimeoutValue

Character expression to return if a timeout occurs (default is empty string).

cCancelValue

Character expression to return if the cancel button is pressed (default is empty string).

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
? INPUTBOX( "Your Text", "Enter Some Text", "Nothing Given", 15000, "Timed Out", "Canceled" )
```

Return Value

Character

Note

The InputBox() dialog displays an edit box and the OK and CANCEL buttons. The OK button returns the TRIM() value of what's in the input box while the cancel button returns the cCancelValue.

If the clnputPrompt is longer than 64 characters, it will be trimmed.

See Also

[GETDATE\(\) Function](#)
[GETDIRECTORY\(\) Function](#)
[GETFILE\(\) Function](#)
[GETTIME\(\) Function](#)
[MESSAGEBOX\(\) Function](#)

INSMODE()

INSMODE([lExpression])

Remarks

Returns the current insert mode and optionally turns the insert mode on or off.

Parameters

lExpression

Logical expression used to turn the insert mode on when true (.T.) or off when false (.F.).

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

IF INSMODE()
    ? "Insert is ON"
ELSE
    ? "Insert is OFF"
ENDIF
```

Return Value

Logical

See Also

CAPSLOCK() Function
NUMLOCK() Function

<https://www.pinvoke.net/default.aspx/user32/GetKeyState.html>

INT()

INT(nExpression)

Remarks

Returns the integer portion of nExpression

Parameters

nExpression

Numeric expression to use with the INT() function.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? INT(PI())
? INT(123.114)
```

Return Value

Numeric

See Also

[CEILING\(\) Function](#)
[FLOOR\(\) Function](#)
[ROUND\(\) Function](#)

INTTOHEX()

INTTOHEX(nExpression)

Remarks

Returns the a character representation of a hexadecimal number equal to the integer portion of the nExpression parameter.

Parameters

nExpression

Numeric expression between 0 and 72,057,594,037,927,936.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? INTTOHEX(256)      && Returns 00FF
? INTTOHEX(192038)    && Returns 0002EE26
```

Return Value

Character

Note

INTTOHEX() begins to lose validity above 72,057,594,037,927,936 as scientific notation will be used at some point above that value.

INTTOHEX() returns either 4, 8, 12, or 16 characters.

See Also

[BINTOC\(\) Function](#)
[CTOBIN\(\) Function](#)
[HEXToint\(\) Function](#)

ISALPHA()

ISALPHA(cExpression [,lAll])

Remarks

Returns a logical true (.T.) if the specified character(s) are between A and Z.

Parameters

cExpression

Character expression to check.

lAll

If false (.F.) or omitted, check the first character, otherwise check all characters.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
? ISALPHA( "A" )           && Returns .T.
? ISALPHA( "ALPHA3" , .T. ) && Returns .F.
```

Return Value

Logical

Note

Setting lAll to true (.T.) will force ISALPHA() to check all characters in the expression. Each character checked must be a letter between A and Z (case insensitive).

See Also

ISBLANK()

ISDIGIT()

ISBLANK()

ISBLANK(eExpression)

Remarks

Determines if an expression is blank or empty.

Parameters

eExpression

JAXBase data type expression.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? ISBLANK( 0 )
? ISBLANK( " " )
? ISBLANK( CHR( 0 ) )
? ISBLANK( { // } )
```

Return Value

Logical

Note

ISBLANK() returns a logical true (.T.) value based on the following.

Expression Type	Evaluates to
Binary or General field Varbinary data	0x0 or zero bytes
Character or Memo	Empty string or all spaces
Currency	0
Date	Value equivalent to CTOD("")
DateTime	Value equivalent to CTOT("")
Double	0
Float	0
Integer	0
Logical	.F.
Numeric	0

See Also

APPEND Command
BLANK Command
EMPTY() Function
ISNULL() Function
LEN() Function

DRAFT

ISDIGIT()

ISDIGIT(cExpression [,cIgnore])

Remarks

Returns a logical true (.T.) if the specified character string are all digits (0 – 9).

Parameters

cExpression

Character expression to check.

cIgnore

Character expression with characters to ignore (default is empty string).

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
? ISALPHA("1234")                                && Returns .T.
? ISALPHA("-1234.17", ".")                      && Returns .F.
? ISALPHA("123-00-9876", "-")                  && Returns .T.
? ISALPHA("(809) 555-1230", "()" "-")        && Returns .T.
```

Return Value

Logical

Note

Any characters found in the cIgnore expression (case sensitive) will be ignored. Each character checked must be a digit between 0 and 9.

ISDIGIT() will is not sufficient to check if the string can be converted to a numeric expression unless cIgnore is empty or omitted.

See Also

[ISBLANK\(\)](#)

[ISDIGIT\(\)](#)

ISEXCLUSIVE()

ISEXCLUSIVE([nWorkArea | cAlias])

Remarks

Returns a logical true (.T.) if the table is opened in exclusive mode.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
USE (_JAXFolder+"data\counties") EXLUSIVE
? ISEXCLUSIVE()
```

Return Value

Logical

Note

A table must be opened using the EXCLUSIVE keyword or while SET EXCLUSIVE is set ON.

Cursors, by their nature, are always opened in EXCLUSIVE mode and ISEXCLUSIVE() will always return a true (.T.) for a cursor.

See Also

[SET EXCLUSIVE Command](#)

[USE Command](#)

ISFLOCKED()

ISFLOCKED([nWorkArea | cAlias])

Remarks

Returns a value of true (.T.) if the table header is locked, otherwise returns a value of false (.F.).

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties") EXLUSIVE
? FLOCKED()  && Returns .T.

SELECT * FROM COUNTIES FOR STATE="WY" INTO CURSOR WYNOT
? FLOCKED("WYNOT")  && Cursor returns .T.
```

Return Value

Logical

Note

Tables opened with the EXCLUSIVE keyword, while EXCLUSIVE is set ON, all cursors, or a table with a lock on the header will all return a value of true (.T.).

If the alias is not found, an error is generated.

See Also

- FLOCK() Function
- ISRLOCKED() Function
- LOCK() Function
- RLOCK() Function
- SELECT SQL Command
- USE Command

ISLEADBYTE()

Status

Slated for Version 2.0

DRAFT

ISLOWER()

`ISLOWER(cExpression [,lAll [,cIgnore]])`

Remarks

Returns a value of true if the checked characters in the string are alpha and lower case.

Parameters

cExpression

Character expression to check.

lAll

Logical expression indicating if all characters should be checked (default is .F.)

cIgnore

Character expression indicating which characters should be ignored (default is empty string).

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? ISLOWER( "aBCD" )          && Returns .T.
? ISLOWER( "aBCD",.T. )       && Returns .F.
? ISLOWER( "alpha-numeric",.T., "-" ) && Returns .T.
```

Return Value

Logical

Note

If a non-alpha character or an upper-case alpha character is found during the check, a value of false (.F.) is returned.

If lAll is true (.T.), then all characters are checked, otherwise just the first character.

If cIgnore is omitted, all characters are checked, otherwise if lAll is .F. then the first character that is not ignored will be checked and if lAll is .T. then all characters that are not in cIgnore are checked.

See Also

[ISALPHA\(\) Function](#)

[ISUPPER\(\) Function](#)

[LOWER\(\) Function](#)

[UPPER\(\) Function](#)

ISNULL()

ISNULL(eExpression)

Remarks

Returns a value of true (.T.) if the expression is .NULL.

Parameters

eExpression

Expression to check

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

eNullValue=.NULL
? ISNULL(eNullValue)      && Returns .T.
? TYPE('eNullValue')     && Returns X
? VARTYPE(eNullValue)    && Returns X
? eNullValue = null       && Returns .NULL.
```

Return Value

Logical

Note

Most JAXBase data types can be set to a null value. If you set a variable holding an object to .NULL., then the object is released and the variable now holds a .NULL. value.

NULL, null, .null., and .NULL. are all equivalent values in an expression.

TYPE and VARTYPE return "X" for null values.

See Also

NVL() Function
SET NULL Command
TYPE() Function
VARTYPE() Function
Null Value Handling

ISODD()

ISODD(nExpression)

Remarks

Returns a value of true (.T.) if the integer portion of a numeric expression is not divided evenly by 2.

Parameters

nExpression

Numeric expression to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? ISODD(2.3)  && Returns .T.
? ISODD(201)  && Returns .F.
```

Return Value

Logical

Note

ISODD() checks the entire integer portion of the expression.

See Also

[INT\(\)](#)

[ROUND\(\)](#)

ISPEN()

Status

Slated for Version 2.0

DRAFT

ISREADONLY()

ISREADONLY([nWorkArea, cAlias])

Remarks

Returns a logical value to indicate if the table is read only.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
USE (_JAXFolder+"data\counties") READONLY
? ISREADONLY()
```

```
SELECT * FROM COUNTIES INTO CUROSOR MyCurs READONLY
? ISREADONLY("MyCurs")
```

Return Value

Logical

Note

A table or cursor must be opened or created with the READONLY keyword.

See Also

[ISEXCLUSIVE\(\) Function](#)

[SELECT Command](#)

[USE Command](#)

ISRLOCKED()

ISRLCOCKED(nRecord [,nWorkArea | cAlias])

Remarks

Returns a true (.T.) value if the record is locked, otherwise returns false (.F.).

Parameters

nRecord

Numeric expression for physical record to check. If nRecord is 0, or if there are no parameters, then it returns the lock status for the current record.

nWorkArea

Numeric expression indicating the work area to check.

cAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties") EXLUSIVE

IF RLOCK()
    ? ISRLOCKED()
ENDIF
```

Return Value

Logical

Note

If nRecord is 0, or if there are no parameters, then it returns the lock status for the current record.

A record lock remains until an UNLOCK is given against it, the table is closed, or JAXBase terminates.

See Also

RLOCK()
FLOCK()
ISFLOCKED()

ISUPPER()

ISUPPER(cExpression [,lAll [,cIgnore]])

Remarks

Returns a value of true if the checked characters in the string are alpha and upper case.

Parameters

cExpression

Character expression to check.

lAll

Logical expression indicating if all characters should be checked (default is .F.)

cIgnore

Character expression indicating which characters should be ignored (default is empty string).

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? ISUPPER( "ABCd" )          && Returns .F.
? ISUPPER( "ABCD" ,.T. )     && Returns .T.
? ISLOWER( "ALPHA-NUMERIC" ,.T. , "-" ) && Returns .T.
```

Return Value

Logical

Note

A non-alpha character or lower-case alpha character found during the check, returns false (.F.).

If lAll is true (.T.), then all characters are checked, otherwise just the first character.

If cIgnore is omitted, all characters are checked, otherwise if lAll is .F. then the first character that is not ignored will be checked and if lAll is .T. then all characters that are not in cIgnore are checked.

See Also

[ISALPHA\(\) Function](#)

[ISUPPER\(\) Function](#)

[LOWER\(\) Function](#)

[UPPER\(\) Function](#)

JSONTOCURSOR()

JSONTOCURSOR(cJSONfile, cCursor)

Status

Slated for Version 2.0

Remarks

Writes JSON data to a cursor.

Parameters

cJSON

Character expression holding the name of the JSON file to convert.

cCursor

Character expression holding the name of the cursor to create or append to.

Return Value

Numeric

Note

CURSORTOJSON() returns the number of records written. If an error occurs during the process, the cursor will not be created, or the cursor being appended to may contain invalid or partial data.

See Also

[CURSORTOJSON\(\) Function](#)

[CURSORTOXML\(\) Function](#)

[XMLTOCURSOR\(\) Function](#)

JSONTOOBJ()

JSONTOOBJ(cJSON)

Remarks

Creates a JAXBase EMPTY class object with properties specified in the JSON string.

Parameters

cJSON

Valid JSON character expression.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
JSONString=[ {"Name": "Fred", "BDate": 1979/07/05, "Balance": 140.73, "LastIn": "2025-04-17 13:41:00"} ]
```

```
oTest=JSONTOOBJ(JSONString)
```

```
DISPLAY MEMORY LIKE oTest
```

Return Value

JAXBase EMPTY Class object

Note

An invalid JSON string will cause an error to be generated.

The object will have property types based on the JSON string of type Array, Charcter, Date, DateTime, Logical, and Numeric.

See Also

[GETJSON\(\) Function](#)

[OBJTOJSON\(\) Function](#)

[PUTJSON\(\) Function](#)

JUSTDRIVE()

JUSTDRIVE(cExpression)

Remarks

Returns the drive or share for Windows and device for Linux from a character expression.

Parameters

cExpression

Character expression with drive-path-file information.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? JUSTDRIVE( "C:\TEMP" )      && Returns "C:" in Windows
```

Return Value

Character

Note

If the cExpression is invalid or does not contain drive information, an empty string is returned.

See Also

[JUSTEXT\(\)](#)
[JUSTFNAME\(\)](#)
[JUSTFULLPATH\(\)](#)
[JUSTPATH\(\)](#)
[JUSTSTEM\(\)](#)

JUSTEXT()

JUSTTEXT(cExpression)

Remarks

Returns the extension from a file name in a character expression.

Parameters

cExpression

Character expression with drive-path-file information.

Example

```
* Demonstrate JUSTTEXT()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET NAMING TO LOWER
? JUSTTEXT( "FILE.TXT" )           && Returns "txt"
? JUSTTEXT( "MyObj.property" )     && Returns "property"
? JUSTTEXT( "c:\TEMP.TEST\Test.jpg" ) && Returns "jpg"
```

Return Value

Character

Note

File name validation is not performed. JUSTTEXT simply strips a file name from the expression and returns anything after the last period. If none is found, an empty string is returned.

SET NAME TO affects JUSTTEXT().

See Also

[JUSTDRIVE\(\)](#)
[JUSTFNAME\(\)](#)
[JUSTFULLPATH\(\)](#)
[JUSTPATH\(\)](#)
[JUSTSTEM\(\)](#)
[SET NAMING Command](#)

JUSTFNAME()

JUSTFNAME(cExpression)

Remarks

Parameters

cExpression

Character expression with drive-path-file information.

Example

```
* Demonstrate JUSTFNAME( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET NAMING TO LOWER
? JUSTFNAME( "FILE.TXT" )           && Returns "file.txt"
? JUSTFNAME( "MyObj.property" )     && Returns "myobj.property"
? JUSTFNAME( "c:\TEMP.TEST\Test.jpg" ) && Returns "test.jpg"
```

Return Value

Character

Note

File name validation is not performed. JUSTFNAME simply strips a file name from the expression and returns it. If none is found, an empty string is returned.

SET NAMING TO affects JUSTFNAME.

See Also

JUSTDRIVE()
 JUSTTEXT()
 JUSTFULLPATH()
 JUSTPATH()
 JUSTSTEM()
 SET NAMING Command

JUSTFULLPATH()

JUSTFULLPATH(cExpression)

Remarks

Returns the path of the file name expression ending with a backslash.

Parameters

cExpression

Character expression with drive-path-file information.

Example

```
* Demonstrate JUSTFULLPATH()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET NAMING TO LOWER
? JUSTFULLPATH ( "FILE.TXT" )
? JUSTFULLPATH ( "c:\TEMP.TEST\" )
? JUSTFULLPATH ( "c:\TEMP.TEST\Test.jpg" )
```

&& Returns ""
&& Returns "c:\temp.test\"
&& Returns "c:\temp.test\"

Return Value

Character

Note

File name validation is not performed. JUSTFULLPATH simply strips a path from the expression and returns it. If none is found, an empty string is returned.

SET NAMING TO affects JUSTFULLPATH.

See Also

[JUSTDRIVE\(\)](#)

[JUSTTEXT\(\)](#)

[JUSTFNAME\(\)](#)

[JUSTPATH\(\)](#)

[JUSTSTEM\(\)](#)

[SET NAMING Command](#)

JUSTPATH()

JUSTPATH(cExpression)

Remarks

Returns the path of the file name expression.

Parameters

cExpression

Character expression with drive-path-file information.

Example

```
* Demonstrate JUSTPATH()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET NAMING TO LOWER
? JUSTPATH("FILE.TXT")
? JUSTPATH ("c:\TEMP.TEST\" )      && Returns ""
? JUSTPATH ("c:\TEMP.TEST\Test.jpg") && Returns "c:\temp.test"
                                         && Returns "c:\temp.test"
```

Return Value

Character

Note

File name validation is not performed. JUSTPATH simply strips a path from the expression and returns it. If none is found, an empty string is returned.

SET NAMING TO affects JUSTPATH().

See Also

[JUSTDRIVE\(\)](#)
[JUSTTEXT\(\)](#)
[JUSTFNAME\(\)](#)
[JUSTFULLPATH\(\)](#)
[JUSTSTEM\(\)](#)
[SET NAMING Command](#)

JUSTSTEM()

JUSTSTEM(cExpression)

Remarks

Returns the stem portion of a file name (filename minus extension).

Parameters

cExpression

Character expression with drive-path-file information.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET NAMING TO LOWER
? JUSTSTEM("FILE.TXT")
? JUSTSTEM ("c:\TEMP.TEST\")
? JUSTSTEM ("c:\TEMP.TEST\Test.jpg")      && Returns "file"
                                            && Returns ""
                                            && Returns "test"
```

Return Value

Character

Note

File name validation is not performed. JUSTSTEM simply strips a file name from the expression, strips the extension from the file name and returns it. If none is found, an empty string is returned.

SET NAMING TO affects JUSTSTEM().

See Also

[JUSTDRIVE\(\)](#)
[JUSTTEXT\(\)](#)
[JUSTFNAME\(\)](#)
[JUSTFULLPATH\(\)](#)
[JUSTPATH\(\)](#)
[SET NAMING Command](#)

KEY()

KEY(nIndexNumber [, nWorkArea | cAlias])

Remarks

Returns the expression used to create the index order.

Parameters

nIndexNumber

Numeric expression indicating which index to interrogate from the index list.

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
SET INDEX TO Counties_ByState
? KEY()
```

Return Value

Character

Note

If nIndexNumber is less than 0, an error is generated, while 0 returns the index currently in control.

If nIndexNumber is greater than the number of indexes open, the empty string is returned.

If no index or alias parameter is given, the current work area is assumed. If an alias is provided and it is not found, an error is generated.

See Also

[DESCENDING\(\) Function](#)

[INDEX Command](#)

[REINDEX Command](#)

[SET INDEX Command](#)

[USE Command](#)

LASTKEY()

LASTKEY()

Remarks

Returns an integer expression corresponding to the last key pressed.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? "Press a key"
? INKEY(0), LASTKEY()
```

Return Value

Numeric

Note

The values returned by LASTKEY() are identical to the values returned by INKEY().

See Also

[INKEY\(\) Function](#)
[READKEY\(\) Function](#)

LEFT()

`LEFT(cExpression, nExpression)`

Remarks

Returns a number of characters starting at the leftmost character.

Parameters

cExpression

Character expression to use to return the leftmost characters.

nExpression

Numeric expression indicating how many characters to return.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
? LEFT( "ABCDEFG" , 4 )
```

Return Value

Character

Note

If there are fewer characters in cExpression than requested by nExpression, the entire expression is returned. If nExpression is 0, the empty string is returned and if nExpression is negative, an error is generated.

See Also

[AT\(\)](#)

[ATC\(\) Function](#)

[ATCLINE\(\) Function](#)

[ATLine\(\) Function](#)

[LTRIM\(\) Function](#)

[RAT\(\) Function](#)

[RATLINE\(\) Function](#)

[RIGHT\(\) Function](#)

[RTRIM\(\) Function](#)

[SUBSTR\(\) Function](#)

LEFTC()

Status

Slated for Version 2.0

DRAFT

LEN()

LEN(cExpression)

Remarks

Returns the number of characters in a character expression.

Parameters

cExpression

Character expression used to return the number of characters.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? LEN( "ABC" )
? LEN( "    Abcdefg    " )
```

Return Value

Character

See Also

[AT\(\)](#)
[ATC\(\) Function](#)
[ATCLINE\(\) Function](#)
[ATLine\(\) Function](#)
[LEFT\(\) Function](#)
[LTRIM\(\) Function](#)
[RAT\(\) Function](#)
[RATLINE\(\) Function](#)
[RIGHT\(\) Function](#)
[RTRIM\(\) Function](#)
[SUBSTR\(\) Function](#)

LENC()

Status

Slated for Version 2.0

DRAFT

LIKE()

`LIKE(cExpression1, cExpression2)`

Remarks

Returns true (.T.) if one character expression matches the other.

Parameters

cExpression1

Compare this expression with cExpression2. cExpression1 may contain wildcards * and ?.

cExpression2

The character expression to compare against.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? LIKE("abc*", "abcdef")      && Returns .T.
? LIKE("abc*f", "abcdef")     && Returns .T.
? LIKE("ab?def", "abcdef")    && Returns .T.
? LIKE("ab?", "abcdef")       && Returns .F.
```

Return Value

Logical

Note

LIKE() returns true (.T.) if cExpression1 matches cExpression2.

Trailing blanks in cExpression1 are used in the comparison.

The wildcard ? matches any single character while * matches any number of characters. You can mix wildcards together in any combination.

See Also

[\\$ Operator](#)

[AT\(\) Function](#)

[ATC\(\) Function](#)

[OCCURS\(\) Function](#)

[RAT\(\) Function](#)

[LIKEC\(\) Function](#)

LIKEC()

Status

Slated for Version 2

DRAFT

LINENO()

LINENO([1])

Remarks

Returns the line number of the currently executing program relative to the main program.

Parameters

1

Returns the line number of the currently executing program relative to the beginning of the current program or procedure.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

ON ERROR DO ErrHandler with LINENO()
ERROR 10
RETURN

PROCEDURE ErrHandler
PARAMETRS tnLineNo
? "Error in line",tnLineNo
return
```

Return Value

Numeric

Note

Usually used in an ON ERROR call to capture the line that threw the error.

See Also

[ERROR\(\) Function](#)
[MESSAGE\(\) Function](#)
[PROGRAM\(\) Function](#)
[Error Messages Listed Numerically](#)

LOADPICTURE()

LOADPICTURE(cFileName,oPictureObject)

Remarks

Loads a picture file into a PICTURE class object.

Parameters

cFileName

Absolute file name to load.

oPictureObject

Picture class object to load to which the file information is loaded.

Example

```
* Demonstrate LOADPICTURE() in a GUI environment
* A text only environment will generate an error
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

oPic=CREATEOBJECT("Picture")
oPic.Left=100
oPic.Right=100
LOADPICTURE(_JAXFolder+"graphics\JAX.jpg", oPic)
oPic.Show()
INKEY(15)
```

Return Value

Logical

Note

LOADPICTURE() provides a cross-platform way to deal with pictures. The picture class object must already exist, and the picture and picture properties will be loaded to the object without modifying any of the other properties (such as position).

LOADPICTURE() returns a value of true (.T.) if the picture loads to the object, otherwise it returns a value of false. The ERRORNO property of LOADPICTURE() gives a JAXBase error result of an error occurs. The ERROR method can be used to create a custom error routine.

See Also

[GETPICTURE\(\) Function](#)

[SAVEPICTURE\(\) Function](#)

[JAXBase Picture Class](#)

LOCK()

LOCK([nWorkArea | cAlias [,nRecord | ArrayName]])

Remarks

Attempts to lock one or more records in a table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

nRecord

Numeric expression of the record to lock

ArrayName

Array containing numeric values of records to lock.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\Counties")
IF LOCK(10)
    ? "Record 10 is locked"
ELSE
    ? "Failed to lock record 10"
ENDIF

INKEY(15000)
UNLOCK ALL
```

Return Value

Logical

Note

Locking even a small number of records will take more time than locking the entire file.

Locking is additive, so if there are records already locked, they will remain locked as you add more to the list.

When passing an array, if even one record fails to lock or a value is greater than the current number of records or less than 0, none of the records will be locked and LOCK() will return a value of false (.F.).

The quickest way to unlock a number of records is to use the UNLOCK ALL command.

SET MULTILOCKS limits locks to one record if set ON. If MULITLOCKS is OFF, then there is no limit.

See Also

CLEAR Commands
CLOSE Commands
ISFLOCKED() Funciton
ISRLOCKED() Function
FLOCK() Function
RLOCK() Function
SET MULTILOCKS Command
SET REPROCESS Command
UNLOCK Command
USE Command

LOG()

LOG(nExpression)

Remarks

Returns the natural logarithm of the specified numeric expression.

Parameters

nExpression

The numeric expression to which you want to get the logarithmic value.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? LOG(1)      && Displays 0.00
? LOG(100)    && Displays 4.61
? LOG(1000)   && Displays 6.91
```

Return Value

Numeric

Note

The base for the natural logarithm is the constant e.

The number of decimal places returned is specified by the SET DECIMALS command.

See Also

[EXP\(\) Function](#)

[LOG10\(\) Function](#)

[SET DECIMALS Command](#)

LOG10()

LOG10(nExpression)

Remarks

Returns the common logarithm value of the specified numeric expression.

Parameters

nExpression

The numeric expression to which you want to get the log.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? LOG10(1)    && Displays 0.00
? LOG10(100)   && Displays 2.00
? LOG10(1000)  && Displays 3.00
```

Return Value

Numeric

Note

The base for the logarithm is 10.

The number of decimal digits displayed is determined by the SET DECIMALS command.

See Also

[EXP\(\) Function](#)

[LOG\(\) Function](#)

[SET DECIM](#)

LOWER()

LOWER(cExpression)

Remarks

Changes the characters in cExpression to all lower case.

Parameters

cExpression

Character expression to change to lower case.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? LOWER( "ALTER EGO" )
```

Return Value

Charcter

See Also

[ISALPHA\(\) Function](#)
[ISLOWER\(\) Function](#)
[ISUPPER\(\) Function](#)
[PROPER\(\) Function](#)
[UPPER\(\) Function](#)

LTRIM()

LTRIM(cExpression)

Remarks

Remove the blank spaces from the left side of the expression.

Parameters

cExpression

Character expression to use in LTRIM()

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? LTRIM( "      HELLO WORLD! " )
```

Return Value

Character

Note

Very useful for removing blank spaces from a numeric expression that was converted to a string by the STR() function.

See Also

ALLTRIM() Function

RTRIM() Function

STR() Function

TRIM() Function

LUPDATE()

```
LUPDATE([nWorkArea | cAlias])
```

Remarks

Returns the date the table was last updated.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\Counties")
? LUPDATE()
```

Return Value

Date

Note

Useful in update procedures. Pulls the date from the table header.

See Also

DIR or DIRECTORY Command

FDATE() Function

FTIME() Function

MAX()

MAX(eExpression1, eExpression2, eExpression3...)

Remarks

MAX returns the maximum value from a list of values of the same type.

Parameters

eExpression1, eExpression2, eExpression3 ...

List of same data type values to compare.

Example

```
* Demonstrate MAX( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? MAX(1,2,3)           && Returns 3
? MAX(-4,1,1.5,0.24,2) ^& Returns 2
? MAX("A", "D4", "p", "Z") && Returns "p"
```

Return Value

Value of same data type as those compared

See Also

CALCULATE Command

MIN() Function

MDY()

MDY(dExpression | tExpression)

Remarks

Returns the date expression in month-day-year format.

Parameters

dExpression

Date expression to return the specified format.

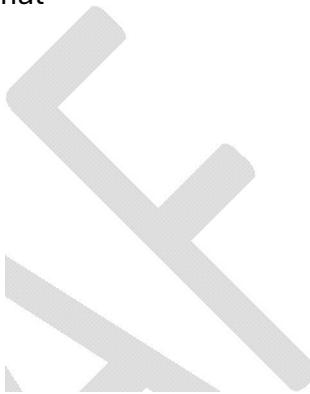
tExpression

DateTime expression to return the specified format

Example

```
* Demonstrate MDY( )
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
SET STRICTDATE TO 0
ACTIVATE CONSOLE
CLEAR

? MDY({1998-05-15}) && Displays May 05, 1998
```



Return Value

Charcter

Note

If SET CENTURY is OFF, the year portion will be two digits, otherwise four digits will be returned.

See Also

DMY() Function

SET CENTURY Command

SET DATE Command

MEMLINES()

MEMLINES(cExpression)

Remarks

Returns the number of lines in a character expression or memo field.

Parameters

cExpression

A character expression or memo field name.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET MEMOWIDTH TO 0
lcString="Fourscore and seven years a
    +"on this continent, a new nat
    +"dedicated to the proposition
    +"Now we are engaged in a great
    +"nation, or any nation so con
    +"long endure.

? MEMLINES(lcString)

SET MEMOWIDTH TO 40
? MEMLINES(lcString)
```

Return Value

Numeric

Note

SET MEMOWIDTH TO 0 means that all lines in a large string or memo field are counted. If MEMOWIDTH is 0 then a CHR(13) character is considered the end of a line. If MEMOWIDTH is set to a positive number, then if the width of a line exceeds the specified value, the line is broken there and or at a CHR(13) character.

See Also

- ALINES() Function
- ATLINE() Function
- COPY MEMO Command
- MLINE() Function
- MODIFY MEMO Command
- SCATTER Command
- SET MEMOWIDTH Command

MEMORY()

MEMORY(nExpression)

Remarks

Returns the amount of memory based on nExpression defining type.

Parameters

nExpression

Numeric expression indicating type of memory to return

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
? MEMORY(1)
```

Return Value

Numeric

Note

The MEMORY() function returns the memory size based on the value defined in the following table.

Value	Description
0 or omitted	Memory required for string (length * 2)
1	Memory required for integer (4 bytes)
2	Memory required for long integer (8 bytes)
3	Memory required for double precision floating point (16 bytes)
4	Memory required for date (8 bytes)

All other values

MESSAGE()

MESSAGE([1])

Remarks

Returns the current JAXBase error message or source line.

Parameters

1

If 1 is given, the current line of source code is returned, otherwise the error text.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
ON ERROR ? MESSAGE()  
ERROR 10
```

Return Value

Character

Note

MESSAGE() is not reset by

See Also

ERROR Command

ERROR() Function

ON ERROR Statement

AERROR() Function

Error Message

MESSAGEBOX()

```
MESSAGEBOX(cMessage [,nDialogBoxType [,cCaption [,nTimeout]]])
```

Remarks

Displays a user-defined dialog box.

Parameters

cMessage

nDialogBoxType

Based on the following tables, the value of nDialogBoxType:

Value	Dialog box buttons
0	OK button only
1	OK and Cancel buttons
2	Abort, Retry, and Ignore buttons
3	Yes, No, and Cancel buttons
4	Yes and No buttons
5	Retry and Cancel buttons

Value	Icon to Display
16	Stop Sign
32	Question Mark
48	Exclamation Point
64	Information Icon

Value	Default Button
0	First button
256	Second button
512	Third button

cCaption

Character string containing the dialog caption.

nTimeout

Numeric value of number of milliseconds to wait for a key before closing the dialog.

Example

```
* Demonstrate MESSAGEBOX()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

MESSAGEBOX("Press the OK button!", 0+48, "Example", 15000)
```

Return Value

Numeric value based on the button selected. Escaping (pressing the Esc key) is the same has hitting the Cancel button.

Value	Button
1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

Note

MESSAGEBOX() requires all four parameters. If you don't want to set a timeout, you need to give a value for all the other parameters.

Setting nTimeout to 0 (default) means the application will wait until the user presses a button is pressed or the user performs a key stroke.

See Also

WAIT Command
INPUTBOX

MIN()

`MIN(eExpression1, eExpression2, eExpression3 ...)`

Remarks

MAX returns the maximum value from a list of values of the same type.

Parameters

`eExpression1, eExpression2, eExpression3 ...`

List of same data type values to compare.

Example

```
* Demonstrate MAX()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? MIN(1,2,3)          && Returns 1
? MIN(-4,1,1.5,0.24,2)  ^& Returns -4
? MIN("A","D4","P","Z")  && Returns "A"
```

Return Value

Value of same data type as those compared.

See Also

CALCULATE Command

MAX() Function

MINUTE()

`MINUTE(tExpression | dExpression)`

Remarks

Returns the minute portion of the specified DateTime value.

Parameters

tExpression

DateTime value to get the MINUTE() value

dExpression

Date value to get the MINUTE() value

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? MINUTE(DATETIME())
```

Return Value

Numeric

Note

The minute portion of the current date and time.

See Also

[CTOT\(\) Function](#)
[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[DTOT\(\) Function](#)
[HOUR\(\) Function](#)
[SEC\(\) Function](#)
[SET SECONDS Command](#)
[TIME\(\) Function](#)

MLINE()

MLINE(cExpression, nLine)

Remarks

Returns the indicated line of a character expression or memo field.

Parameters

cExpression

Character expression or memo field used extract the indicated line.

nLine

Numeric expression indicating which line to extract.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET MEMOWIDTH TO 0
lcString="Fourscore and
    +"on this contin
    +"dedicated to t
    +"Now we are eng
    +"nation, or any
    +"long endure."
? MLINE(2)
```

Return Value

Character

Note

SET MEMOWIDTH specifies the width of each line to be calculated. If MEMOWIDTH is set to 0 then a line terminates at a CHR(13). If MEMOWIDTH is greater than 0, then lines terminate either after that many characters, or at a CHR(13).

See Also

[ALINES\(\) Function](#)

[ATLINE\(\) Function](#)

[MEMLINES\(\) Function](#)

[SET MEMOWIDTH Command](#)

MOD()

MOD(nDividend, nDivisor)

Remarks

Returns the modulus; the remainder of the nDividend divided by the nDivisor.

Parameters

nDividend

The numeric expression to divide.

nDivisor

The numeric expression of how many times to divide nD

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? MOD(51,7)  && Returns 2
? MOD(11,10) && Returns 1
? MOD(15.7/5) && Returns 0.2
```

Return Value

Numeric

Note

The MOD() function returns the same results as the % operator.

See Also

% Operator

MONTH()

MONTH(dExpression | tExpression)

Remarks

Returns the month number from a Date or DateTime expression.

Parameters

dExpression

Date expression used to extract the month.

tExpression

DateTime expression used to extract the month.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
SET STRICTDATE TO 0
SET DATE TO AMERICAN
ACTIVATE CONSOLE
CLEAR

? MONTH({12/14/2025})
? MONTH({2023/7/1})
```

Return Value

Numeric

Note

MONTH() returns the month number. January is month 1 and December is the 12th month.

See Also

[CMONTH\(\)](#)
[DAY\(\) Function](#)
[QUARTER\(\) Function](#)
[YEAR\(\) Function](#)

NDX()

`NDX(nIndex [,nWorkArea | cAlias])`

Remarks

Returns the name of the specified index in the current or specified work area or alias.

Parameters

nIndex

Numeric expression indicating what index name to return.

nWorkArea

Numeric expression that indicates which work area to use.

cAlias

Character expression that indicates which alias to use.

Example

```
* Demonstrate NDX()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\c
SET INDEX TO Counties_B
? NDX(1)
```

Return Value

Character

Note

If nIndex does not identify an index, an empty character value is returned.

If nIndex refers to an index file (Slated for Version 2.0) then the name will be returned in the format CDXName where CDX is the file extension and XName is the index name. If it is an IDX file, just the file name stem of the IDX file is returned.

See Also

[INDEX Command](#)

[KEY\(\) Function](#)

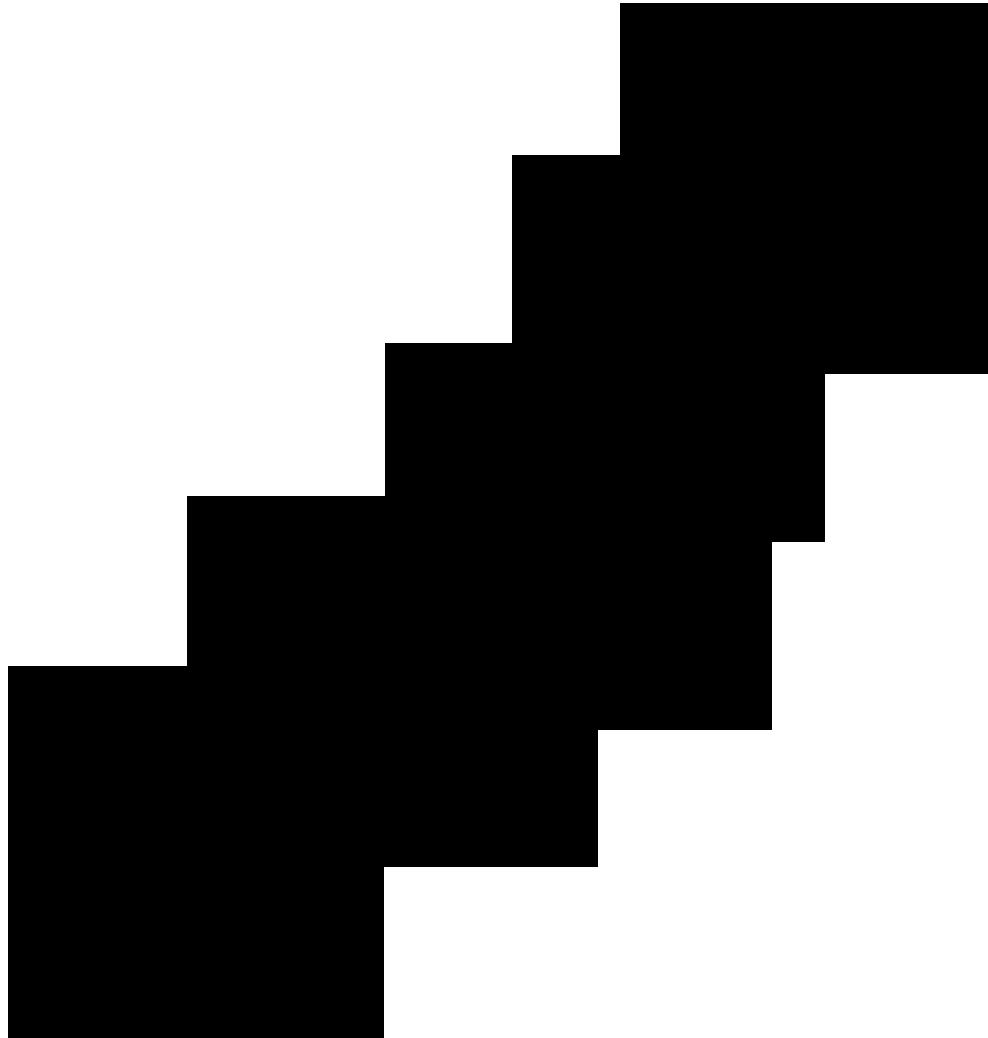
[SET INDEX Command](#)

[USE Command](#)

NORMALIZE()

Status

Slated for Version 2.0



NUMLOCK()

NUMLOCK([lExpression])

Remarks

Returns the current state of the Num Lock key and optionally sets the state.

Parameters

lExpression

Logical expression that, if it is present, will set the state

Example

```
* Demonstrate NUMLOCK()
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

IF NUMLOCK()
    ? "NUMLOCK is set on"
ELSE
    ? "NUMLOCK is set off"
ENDIF
```

Return Value

Logical

See Also

CAPSLOCK

INSMODE()

NVL()

NVL(eExpression1, eExpression2)

Remarks

Returns a non-null value from two expressions.

Parameters

eExpression1, eExpression2

NVL() returns eExpression2 if eExpression1 is null, otherwise it returns eExpression1.

Example

```
* Demonstrate NVL  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
mNul=.NULL.  
? NVL(mNul, "Not Null")
```

Return Value

JAXBase data type or object

Note

Use the NVL() function to return the value of eExpression1 where it is not supported or not relevant.

See Also

ISNULL() Function
SET NULL Command

OBJTOCLIENT()

AOBJTOCLIENT(ArrayName, oObject)

Remarks

Creates an array containing position information for the specified object.

Parameters

oObject

Object on which to return position information.

Example

```
* Demonstrate OBJTOCLIENT
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

Frml=CREATEOBJECT( "FORM" )
Frml.Height=100
Frml.Width=200

Frml.ADDOBJECT("lblTest1","label")
Frml.Caption="Hello"
Frml.lblTest.Left=30
Frml.lblTest.Top=40

? CLIENTTOOBJ(aClient,F
LIST MEMORY LIKE aClient
```

Return Value

Numeric

Note

If ArrayName is not specified, it is created. If it is specified, otherwise it is overwritten with the correct number of corresponding elements.

Each array element contains an array with the following properties.

Property	Data Type	Description
BaseClass	Character	Baseclass of object
Height	Numeric	Height of object
Left	Numeric	Pixels oObject is from the left edge
Name	Character	Name of object
Object	Object	Object Reference
Top	Numeric	Pixels oObject is from top of object
Width	Numeric	Width of object

The first row will contain the oObject with zero as Left and Width. Subsequent rows will contain the hierarchy of objects that contain oObject, and the left and width values indicate where oObject is in relation to each object in the hierarchy.

As an example: If you create a form with a container that holds a pageframe and an object that is placed on a page, the OBJTOCLIENT() array will contain 5 rows. The first row will be for oObject, the second row will contain the page information, the third row will contain the pageframe information, the fourth row will contain the container information, and the fifth row will contain the form information. If the form is part of a form set, there will be an additional row for the formset, but all numeric properties for a formset are always zero as it is

See Also

[Height Property](#)
[Left Property](#)
[Name Property](#)
[Top Property](#)
[Width Property](#)

OBJTOJSON()

OBJTOJSON(oObject[,cControl])

Remarks

Creates a JSON string of an object's properties.

Parameters

oObject

JAXBase class object

cControl

Character expression providing control to what is saved.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
SCATTER NAME oRow
? OBJTOJSON(oRow)
? OBJTOJSON(oRow, "LIKE :")
? OBJTOJSON(oRow, "EXCEPT :")
```

Return Value

Character

Note

OBJTOJSON() creates a JSON string representing the object's properties base on the contents of the object.

The optional parameter cControl is a character string that begins with the word LIKE: or EXCEPT: and is then followed by one or more field name skeletons, separated by commas, indicating whether to include or exclude properties with those names.

See Also

[GETJSON\(\)](#)
[JSontoObj\(\)](#)
[PutJSON\(\)](#)

OCCURS()

OCCURS (cSearchExpression, cExpressionToSearch)

Remarks

Returns the number of times a string occurs in another string.

Parameters

cSearchExpression

Character expression of string to search for in cExpression

cExpressionToSearch

Character expression to use in searching for cSearchExp

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
lcString="Fourscore and seven years ago our fathers brought forth  
+ "on this continent a new nation, conceived in liberty and  
+ dedicated to the proposition that all men are created equal.  
+ "Now we are engaged in a great civil war, testing whether  
+ "nation, or any nation so conceived and so dedicated, can long endure."
```

? OCCURS ("f")
? OCCURS ("1")

Return Val

Numeric

Note

OCCURS()

OCCURS() [REDACTED] condition isn't found in cExpressionToSearch.

See Also

\$ Operator

AT() Function

ATLINE() Function

RAT() Function

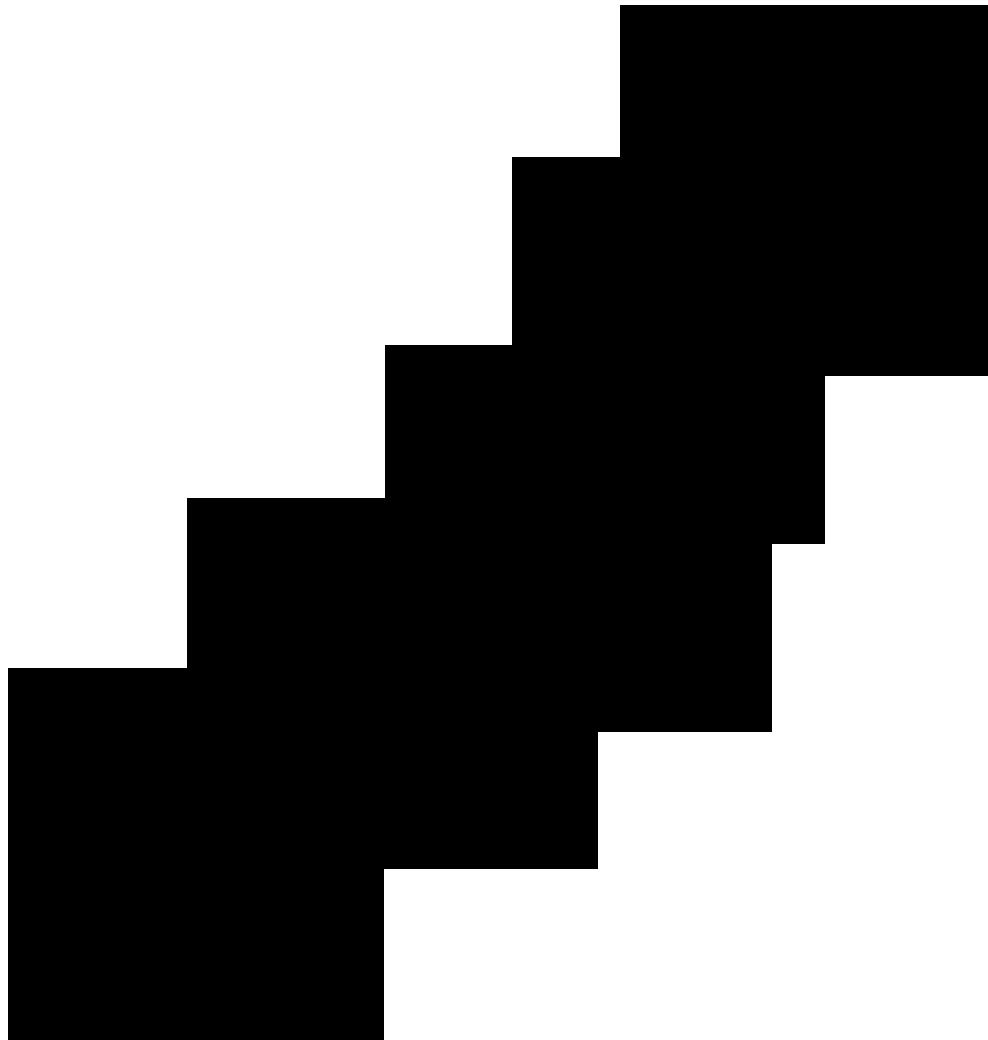
RATLINE() Function

INLIST() Function

OLDVAL()

Status

Slated for Version 2.0



ON()

`ON(cCommand [,LabelName])`

Remarks

Returns the command assigned to the event-handling commands:

- ON ERROR
- ON ESCAPE
- ON KEY LABEL

Parameters

cCommand

Character expression that specifies one of the even-handlers.

cCommand	ON Command	
ERROR	ON ERROR	
ESCAPE	ON ESCAPE	
KEY	ON KEY	

To return the command currently reserved for the error condition:
`? ON("ERROR")`

LabelName

Used with the ON KEY LABEL command to identify the key combination that is assigned a command.

For a complete list of key combinations, see the Key Codes topic.

To return the command currently reserved for the key F12:
`? ON("KEY", "F12")`

Example

```
* Demonstrates the use of the ON command.
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE MODE
SET CONSOLIDATE MODE
ACTIVATE COMMAND WINDOW
CLEAR

* Assign to the key which will be
* available after the program ends
ON KEY LABEL F12 ? "HELLO WORLD!"

? ON("KEY", "F12")
```

Return Value

Character

Note

This function is most often used when a module needs to turn off or reassign an event command for a short period of time. It will save the current value of the event, assign it a new value, and when it's done, reassign the old value back.

This is especially useful when you want to have a local error handler designed specifically for the code that is about to be run.

If there is no command assigned to the specified event handler, ON() returns the empty string.

See Also

[INKEY\(\) Function](#)

[LASTKEY\(\) Function](#)

[ON ERROR Command](#)

[ON ESCAPE Command](#)

[ON KEY Command](#)

[ON KEY LABEL Command](#)

ORDER()

```
ORDER([nWorkArea | cAlias [, nSwitch]])
```

Remarks

Returns the name of the controlling index for the specified table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cAlias

Character expression indicating the alias of the table to

nSwitch

Numeric value that dictates what is returned in nWorkArea.

Value	Description
0 or omitted	Controlling index name
1	Controlling index file name
2	.T. if controlling index o

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFDB)
SET INDEX TBL1 ON
? ORDER()
? ORDER(0,1)
? ORDER(0,2)
```

DESCENDING keyword

Return Value

Character expression

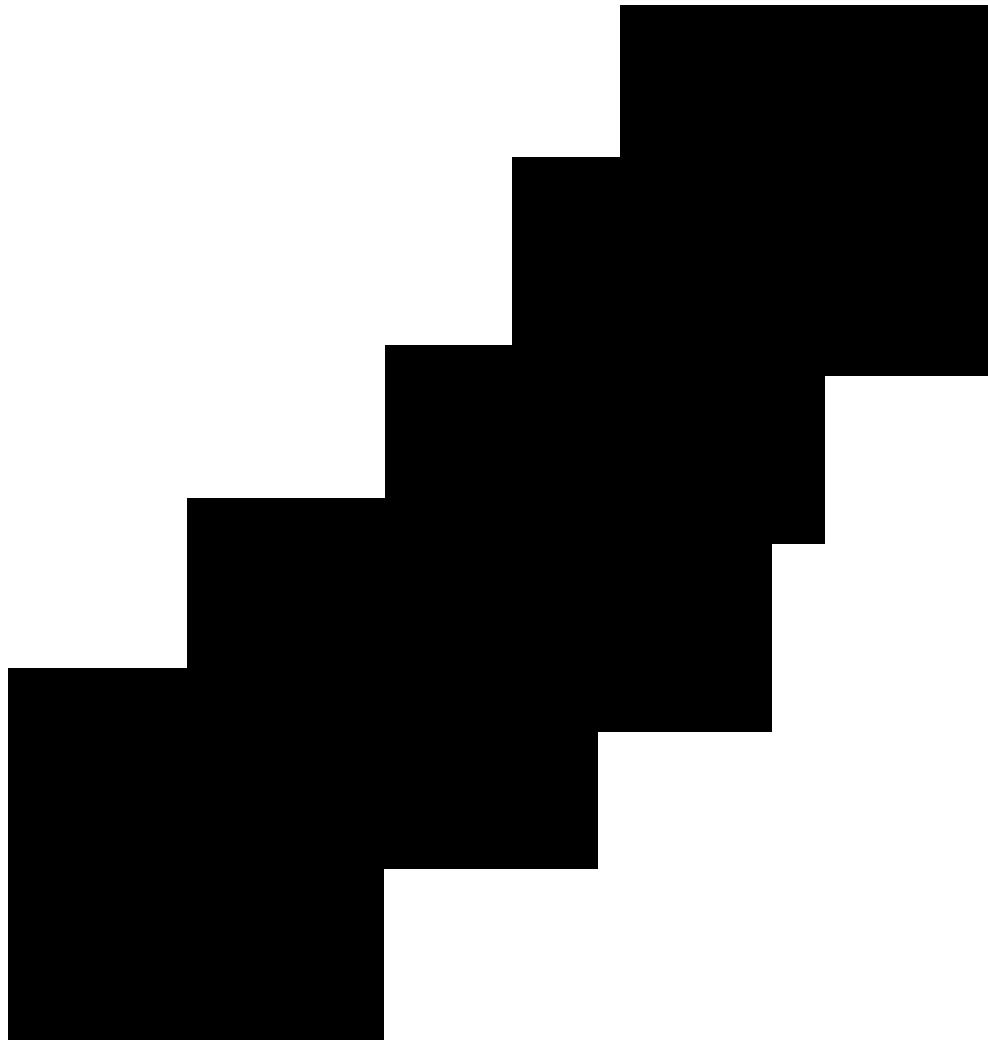
Note

Tables often have several indexes open at the same time. ORDER() allows you to know which one is in control, where it's located, and if it's set for DESCENDING order. nSwitch value of 2 does not care if the index was created with a descending order, only if it was opened with the DESCENDING keyword.

A nWorkArea value of 0 means the current work area.

See Also

DESCENDING() Function
INDEX Command
SET INDEX Command
SET ORDER Command
USE Command



OS()

OS()

Remarks

Returns the operating system type

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? OS() && Returns WINDOWS or LINUX
```

Return Value

Character

See Also

GETENV()

_OS object

PADC() / PADL() / PADR()

```
PADC(eExpression, nResultSize [, cPadCharacter])
PADL(eExpression, nResultSize [, cPadCharacter])
PADR(eExpression, nResultSize [, cPadCharacter])
```

Remarks

Returns a string from an expression, padded with spaces or specified characters to the specified length, causing the eExpression to be centered, left justified, or right justified in the string.

Parameters

eExpression

Expression to convert to a string.

nResultSize

Length of the resulting string.

cPadCharacter

Character to use for padding. If omitted

Example

```
* Demonstrate PADC(), PADL(), PADR()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? PADC("HELLO", 20, " ")
? PADL("HELLO", 20, " ")
? PADR("HELLO", 20, " ")
```

Return Value

Character

Note

PADC() centers the string, PADL() left justifies the string by putting the pad characters on the left, and PADR() right justifies the string by putting the pad characters on the right.

See Also

[ALLTRIM\(\) Function](#)

[LEFT\(\) Function](#)

[LTRIM\(\) Function](#)

[RIGHT\(\) Function](#)

[STUFF\(\) Function](#)

[TRIM\(\) Function](#)

PARAMETERS()

PARAMETERS()

Remarks

Returns the number of parameters most recently passed to a program, procedure, method, or user defined function.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
DO TEST1 WITH 1,5,.T.,"HI"  
RETURN  
  
PROCEDURE TEST1  
APARAMETERS aParms  
? PARAMETERS( )  
RETURN
```

Return Value

Numeric

Note

PARAMETERS() is useful for decisions on what to do based on the number of parameters passed, allowing error checking and default values to be set.

See Also

APARAMETERS
DO Command
FUNCTION
LPARAMETERS
PARAMETERS
PROCEDURE

PAYMENT()

`PAYMENT(nPrincipal, nInterestRate, nPayments)`

Remarks

Returns the amount of each payment on a fixed-interest loan.

Parameters

`nPrincipal`

Numeric expression specifying the original loan amount.

`nInterestRate`

Numeric expression specifying the interest rate per period. If you specify annual interest, divide the interest rate by 12.

`nPayments`

Numeric expression specifying the total number of payments.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
? PAYMENT(100000, .105/12, 360)
```

Return Value

Numeric

Note

`PAYMENT()` provides a way to calculate monthly payments on a loan with constant principal and fixed interest providing the ability to create a payment plan.

See Also

`CALCUALT`

`FV() Function`

`PV() Function`

PEMSTATUS()

`PEMSTATUS(oObject, cName)`

Status

Slated for Version 2.0

Remarks

Returns information on a property, event, or method in an object.

Parameters

`cObject`

Object to use.

`cName`

Character expression containing the name of the property, event, or method to get information.

Return Value

Object

Note

PEMSTATUS returns an object.

Property	Data Type	Description
Changed	Logical	True if changed since instantiation
Inherited	Logical	True if inherited from parent class
Name	Character	Name of the property, event, or method
Protected	Logical	True if protected
ReadOnly	Logical	True if read-only
Type	Character	Type of the property, event, or method
UserDefined	Logical	True (.T.) if user-defined

PI()

`PI([nDecimals])`

Remarks

Returns the value of PI

Parameters

`nDecimals`

Numeric expression 0 to 15 indicating how many decimal places.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? PI()  
? PI(4)  
? PI(10)
```

Return Value

Numeric

Note

The number of digits returned is limited by the system's floating point precision and the nDecimals parameter, which is limited to 15.

See Also

`SET DECIMALS`

PROGRAM()

Remarks

Parameters

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Return Value

Note

See Also

PROPER()

PROPER(cExpression)

Remarks

Changes the characters in cExpression to proper case

Parameters

cExpression

Character expression to change to proper case where the first character is capitalized, as in a person's name.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? PROPER( "ALTER EGO" )
```

Return Value

Charcter

See Also

[ISALPHA\(\) Function](#)
[ISLOWER\(\) Function](#)
[ISUPPER\(\) Function](#)
[LOWER\(\) Function](#)
[UPPER\(\) Function](#)

PUTJSON()

`PUTJSON(cJSON, cName, eExpression)`

Remarks

Adds or updates a field name and value to a JSON string.

Parameters

cJSON

JSON string to update.

cName

Field name to add or update.

eExpression

JAXBase expression to place into the JSON string.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
SET STRICTDATE TO 0  
ACTIVATE CONSOLE  
CLEAR
```

```
cJSON=PUTJSON( " " , "Name"  
cJSON=PUTJSON( cJSON , "BD"  
cJSON=PUTJS
```

```
DIMENSION a  
aTest[1]=7  
aTest[2]=1  
aTest[3]=2
```

```
cJSON=PUTJS
```

```
? cJSON
```

Return Value

Character

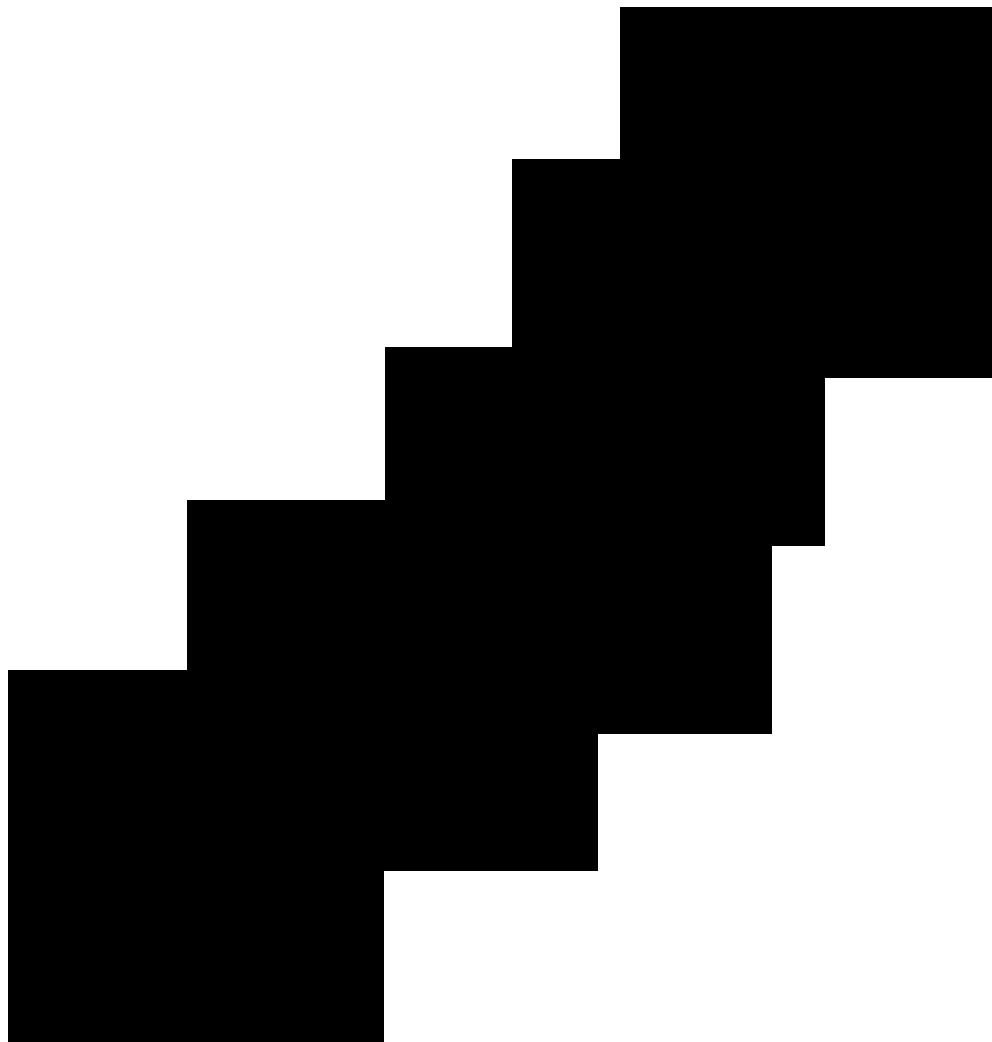
Note

PUTJSON() will accept an empty string and create a valid JSON string with the property provided.

PUTJSON() can add any JAXBase data type or object, though care must be taken with objects and arrays, otherwise the JSON string can become quite long.

See Also

GETJSON()
JSONTOOBJ()
OBJTOJSON()
PUTJSON()



PUTFILE()

Remarks

Parameters

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Return Value

Note

See Also

PV()

PV(nPayment, nInterestRate, nPayments)

Remarks

Returns the present value amount of each payment (positive or negative).

Parameters

nPayment

Numeric expression specifying the periodic payment amount.

nInterestRate

Numeric expression specifying the interest rate per period. If you know the annual interest, divide the interest rate by 12.

nPayments

Numeric expression specifying the total number of payments.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
? PV(500, .075/12, 48)
```

Return Value

Numeric

Note

PV() assumes a constant interest rate over time. This function provides the ability to create a investment plan.

See Also

CALCUALT

FV() Functi

PAYMENTS() Function

QUARTER()

```
QUARTER(dExpression | tExpression [, nMonth])
```

Remarks

Returns the quarter of the year in which a date or datetime expression occurs.

Parameters

dExpression

Date expression to get the quarter.

tExpression

DateTime expression to get the quarter.

nMonth

Starting month for the first quarter so you can base the quarter on a calendar year.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
? QUARTER(DATE())
```

Return Value

Numeric

Note

Returns 0 indicating first quarter, 1 – 3 indicating second quarter, 4 – 6 indicating third quarter, and 7 – 9 indicating fourth quarter.

See Also

CMONTH()

DATE() Function

DATETIME()

RAND()

RAND(nSeedValue)

Remarks

Returns a random number between 0 and 1.

Parameters

nSeedValue

Numeric expression specifying the seed value which determines the sequence.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

* Get a random value from 1 to 100
? RAND(-1)
FOR I=1 TO 100
    ? INT(RAND()*10)+1
ENDFOR
```

Return Value

Numeric

Note

RAND() returns the same value each time it is called followed by subsequent RAND() calls without the seed value specified.

If nSeedValue is not specified, the current value of the system clock is used.

The system clock is initialized to the value 17,760,704 when the application initializes.

RAT()

`RAT(cSearchExpression, cExpressionSearched [, nOccurrence])`

Remarks

Search a character expression or memo field for the last occurrence of another character expression.

Parameters

cSearchExpression

The character expression to be used in the search.

cExpressionToSearch

The character expression to be used in the search. The expression can be a character field or memo field.

nOccurrence

Which occurrence to return. If omitted, 1 is assumed. If omitted or < 1, returns 0.

Example

```
* Demonstrate ASCAN()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

aStr="FOUR SCORE"
? RAT("O",aStr)      &&
? RAT("O",aStr,2)
? RAT("s",aStr,3)
```

Return Value

Numeric

Note

RAT() is case sensitive.

If the indicated occurrence of cSearchExpression or occurrence of same is not found, then 0 is returned.

The value returned indicates the starting position of the first character of the matched cSearchExpression.

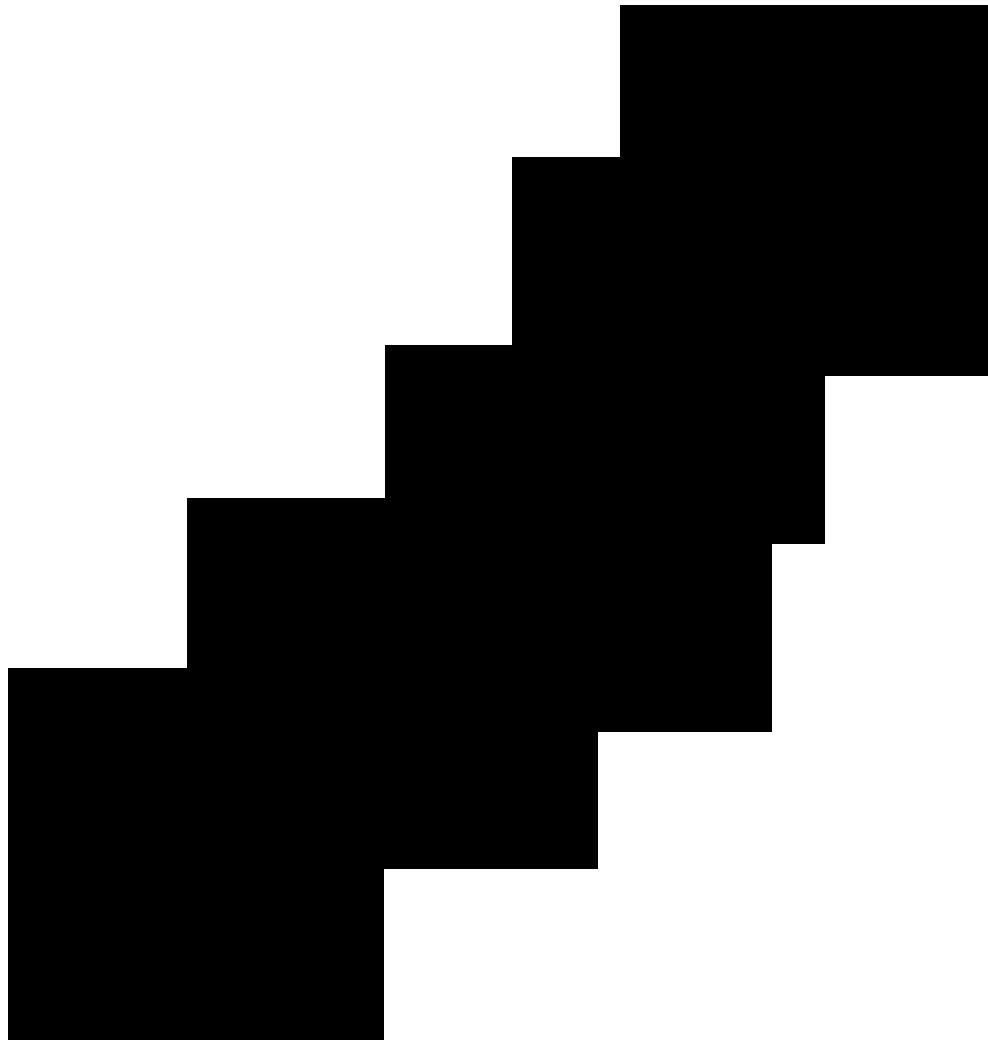
See Also

\$ Operator
AT() Function
AT_C() Function
ATCC() Function
ATLINE() Function
LEFT() Function
RATLINE() Function
RIGHT() Function
SUBSTR() Function
SUBSTRC() Function
STREXTRACT() Function
OCCURS() Function
INLIST() Function

RATC()

Status

Slated for Version 2.0



RATLINE()

RATLINE(cSearchExpression, cExpressionToSearch, [,nOccurrence] [, cParseCharacter])

Remarks

Search an expression for the last matching expression returning which line it was found.

Parameters

cSearchExpression

The character expression to be used in the search.

cExpressionToSearch

The character expression to be used in the search. The expression can be a character field or memo field.

nOccurrence

Which occurrence to return. If omitted, 1 is assumed. If omitted, 0 is returned.

cParseCharacter

The character to use to terminate a line. If omitted, the carriage return character is used. When the SET MEMOWIDTH value is used. If using SET MEMOWIDTH, the carriage return character is always returned for a match.

Example

```
* Demonstrate ATLINE()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE

lcString="I have a dream that one day my people will be free.
          brought forth, "+CHR(13)
          + "one nation under God, indivisible, with
          + "a new birth of freedom and
          + "new hope for our country and for all
          + "men everywhere who seek
          + "the blessings of liberty and justice for
          + "all men." +CHR(13)
          + "We must not let ourleanor
          + "dedicated, can "+CHR(13)

? RATLINE(lcString, "one", 2)
? RATLINE(lcString, "one", 1)
? RATLINE(lcString, "one", 0)
```

&& Returns 2
&& Returns 1
&& Returns 0

Return Value

Numeric

Note

RATLINE() is a case sensitive search.

If the search is successful, the line where the match was found is returned, otherwise 0 is returned.

If using SET MEMOWIDTH and its value is set to 0 then 1 is always returned for a match.

See Also

\$ Operator
AT() Function
ATC() Function
ATCLINE() Function
ATLINE() Function
INLIST() Function
LEFT() Function
MLINE() Function
OCCURS() Function
RAT() Function
RIGHT() Function
SET MEMOWIDTH Command
SUBSTR() Function

RECCOUNT()

`RECCOUNT([nWorkArea | cAlias])`

Remarks

Returns the number of records in the specified table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
USE (_JAXFolder+"data\counties")  
? RECCOUNT()
```

Return Value

Numeric

Note

If no parameter is provided, the current work area is used. If an alias is provided and not found, an error is generated.

See Also

[FSIZE\(\) Function](#)
[HEADER\(\) Function](#)
[RECNO\(\) Function](#)
[RECSIZE\(\) Function](#)

RECNO()

RECNO([nWorkArea | cAlias])

Remarks

Returns the physical record under the current record pointer.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties")
LOCATE FOR STATE="MN"
? RECNO()
```

Return Value

Numeric

Note

If no parameter is provided, the current work area is used. If an alias is provided and not found, an error is generated.

See Also

FSIZE() Function

HEADER() Function

RECCOUNT() Function

RECSIZE() Function

RECSIZE()

RECSIZE([nWorkArea | cAlias])

Remarks

Returns the size of the record in the specified DBF table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
USE (_JAXFolder+"data\counties")  
? RECSIZE()
```

Return Value

Numeric

Note

If no parameter is provided, the current work area is used. If an alias is provided and not found, an error is generated.

See Also

[FSIZE\(\) Function](#)

[HEADER\(\) Function](#)

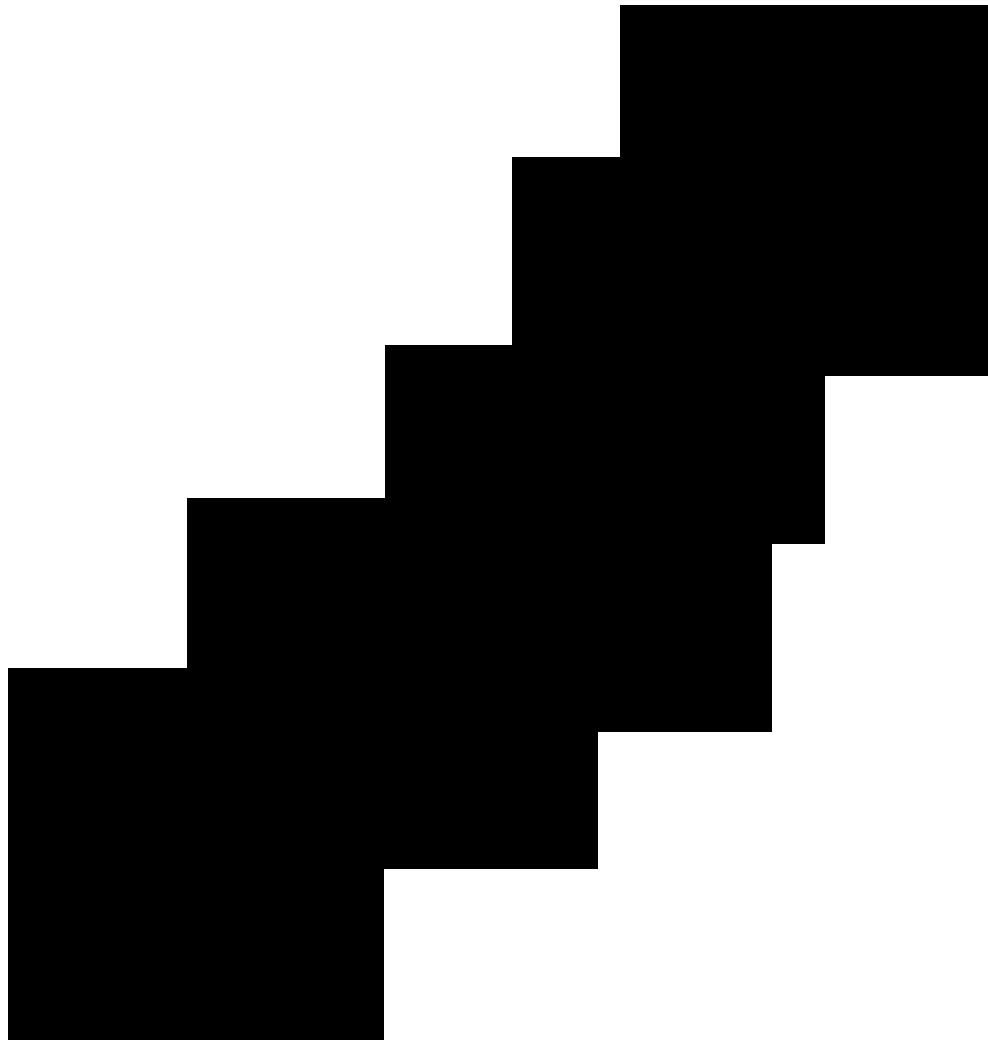
[RECCOUNT\(\) Function](#)

[RECNO\(\) Function](#)

RELATION()

Status

Slated for Version 2.0



REMOVEPROPERTY()

REMOVEPROPERTY(oObject, cPropertyName)

Remarks

Removes a non-native property from an object.

Parameters

oObject

JAXBase object containing the property you wish to remove.

cPropertyName

Character expression containing the name of the property.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
USE (_JAXFolder+"data\counties")  
SCATTER NAME oRow  
DISPLAY MEMO LIKE oRow  
  
REMOVEPROPERTY(oRow, "name")  
DISPLAY MEMO LIKE oRow
```

Return Value

Logical

Note

REMOVEPROPERTY() returns TRUE if the property was successfully removed from the object.

You cannot remove native properties from an object, such as trying to remove the name or baseclass property from an object.

See Also

[ADDPROPERTY\(\) Function](#)

[AddProperty Method](#)

REPLICATE()

REPLICATE(cExpression, nTimes)

Remarks

Returns a string of specified characters repeated a specified number of times.

Parameters

cExpression

Character expression containing one or more characters.

nTimes

Numeric expression containing the number of times to repeat the character expression.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? REPLICATE( "=====", 10 )
```

Return Value

Character

Note

The maximum length of the returned string is limited by available memory.

See Also

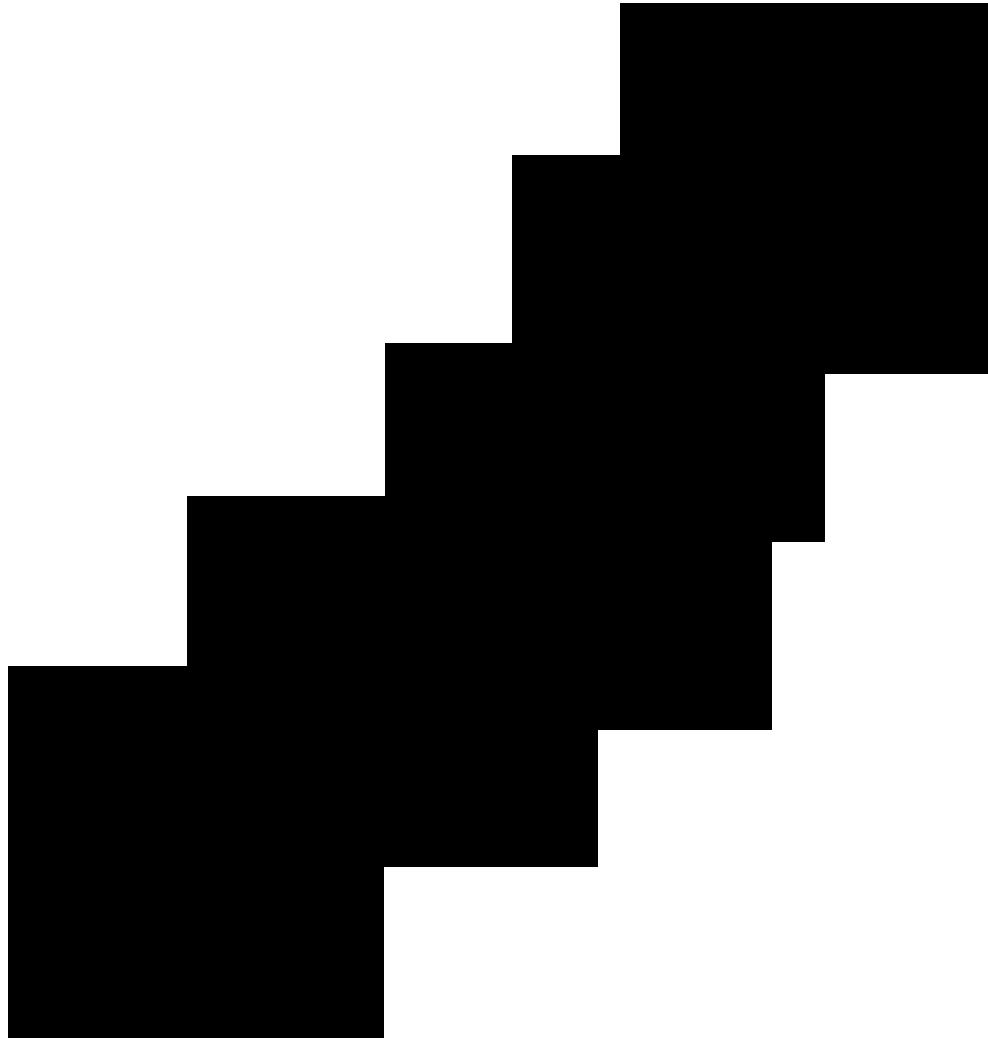
SPACE() Function

CLEAR Console

REQUERY()

Status

Slated for Version 2



RGB()

RGB(nRec, nGreen, nBlue)

Remarks

Returns an integer value based on red/green/blue values.

Parameters

nRed

Numeric expression from 0 to 255 indicating intensity of red.

nGreen

Numeric expression from 0 to 255 indicating intensity of green.

nBlue

Numeric expression from 0 to 255 indicating intensity of blue.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

oForm=CREATEOBJECT( "FORM" )
oForm.Show
=INKY(5)
oForm.BackColor=RGB(0,0,0)
=INKY(5)
oForm.BackColor=RGB(255,255,255)
=INKY(5)
oForm.BackColor=RGB(0,0,255)
=INKY(5)
oForm.BackColor=RGB(255,0,0)
=INKY(5)
RELEASE oForm
```



Return Value

Numeric

Note

The value created by RGB() can be used to set color properties in objects.

See Also

[GETCOLOR\(\) Function](#)

[BackColor Property](#)

[ForeColor Property](#)

RIGHT()

RIGHT(cExpression, nExpression)

Remarks

Returns the specified number of rightmost characters.

Parameters

cExpression

Character expression containing string which you want to extract characters from.

nExpression

Numeric expression indicating how many characters to extract.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? RIGHT( "HELLO WORLD!" , 6 )  && Returns
```

Return Value

Character

Note

If nExpression is greater than the number of characters in cExpression, the function returns all characters in cExpression.

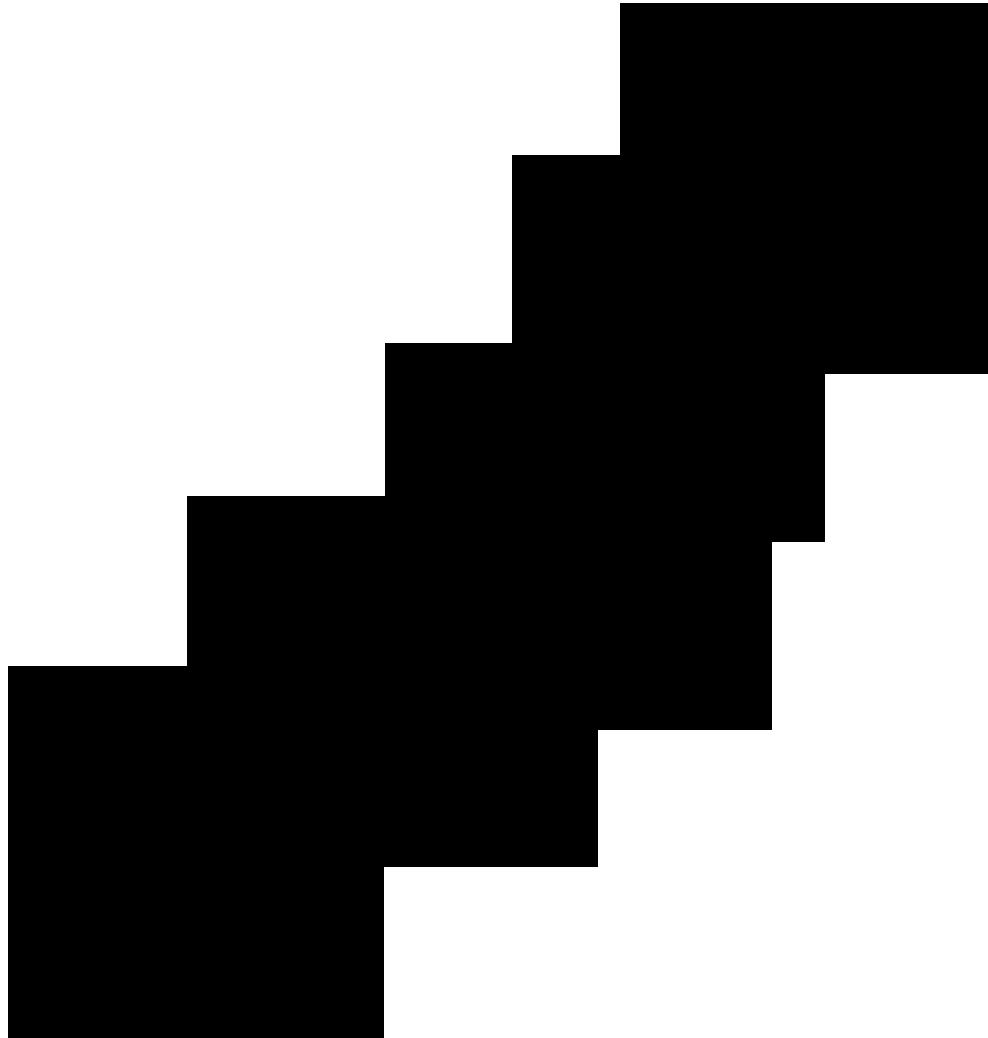
See Also

[AT\(\) Function](#)
[ATLINE\(\) Function](#)
[LEFT\(\) Function](#)
[LTRIM\(\) Function](#)
[RTRIM\(\) Function](#)
[SUBSTR\(\) Function](#)
[TRIM\(\) Function](#)

RIGHTC()

Status

Slated for Version 2.0



RLOCK()

RLOCK([nWorkArea | cAlias [, cRecordList | nRecord | Array]])

Remarks

Attempt to lock records in a table.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to

cRecordList

Character expression holding one or more recor

nRecord

Numeric expression holding the record

Array

An array of numbers which correspond

Example

* Demonstrate

CLEAR ALL

CLOSE ALL

SET EXCLUSI

SET CONSOL

ACTIVATE CO

CLEAR

USE (_JAXF

IF RLOCK(0

 ? "S

ELSE

 ? "F

ENDIF

Return Value

Logical

Note

If multiple records are being locked, then all must successfully lock, otherwise none will lock and a value of false (.F.) will be returned.

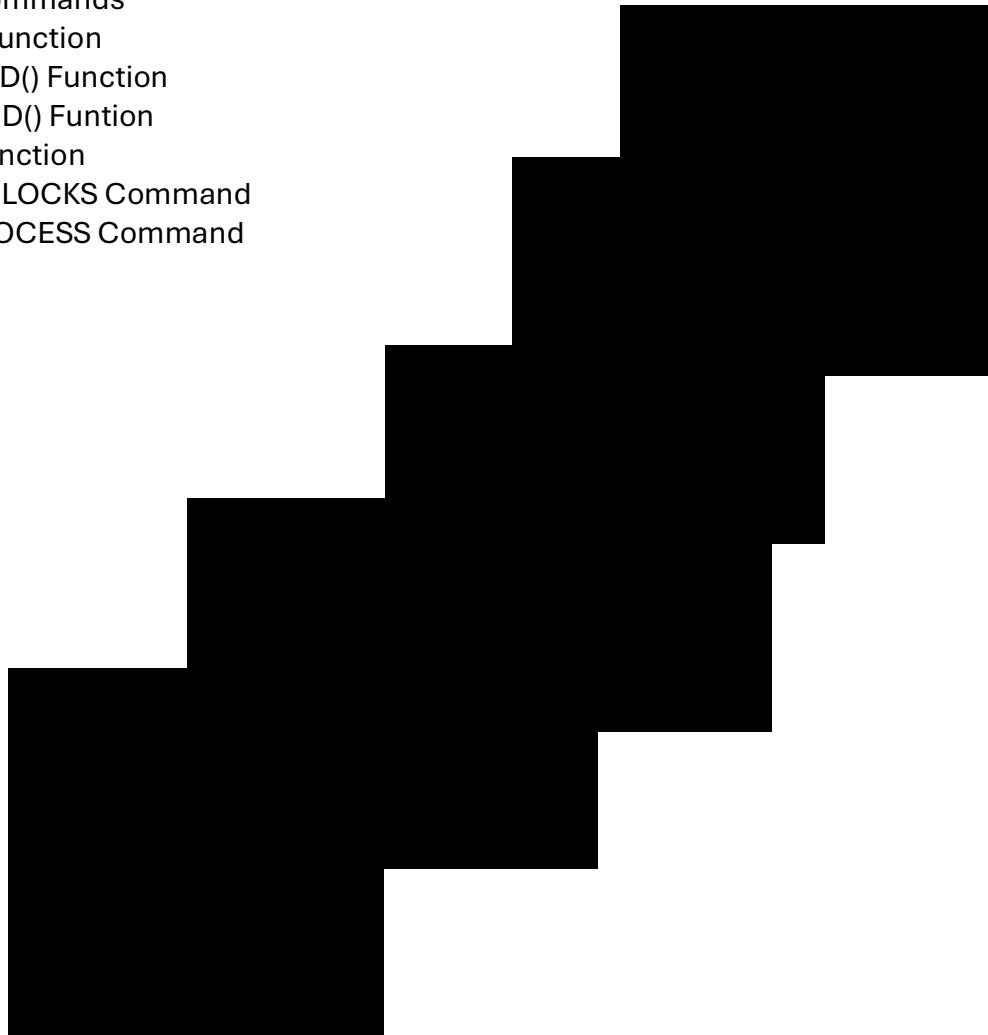
SET MULTILOCKS affects how many records you can lock. If set ON then you can only lock one record.

A lock stays in effect until released, the table is closed, or JAXBase terminates.

A lock can only be released by the user who sets it.

See Also

CLEAR Commands
CLOSE Commands
FLOCK() Function
ISFLOCKED() Function
ISRLOCKED() Function
LOCK() Function
SET MULTILOCKS Command
SET REPROCESS Command



ROLLBACK()

ROLLBACK()

Remarks

Rollback the current SQL transaction.

Example

```
* Interrogate the JAXCorp database that you installed
* onto your MySQL or SQL Server database server using
* the CreateDSNFile program in the TOOLS folder.
```

```
CLEAR ALL
CLOSE ALL
CLEAR
```

```
SET NAMING TO LOWER
OPEN DATABASE JAXCorp USING DECRYPT(FILETOSTR("C:\JAXBase\JAXCorp.mdb", "snx" ))
```

```
BEGIN TRANSACTION
    JAXCorp.Exec("Insert into log (event) values ('Test Transaction')") ? "Success"
    IF COMMIT()
        ? "Committed"
    ELSE
        * Failed to commit, so roll back
        IF ROLLBACK()
            ? "Rolled back"
        ELSE
            ?
        ENDIF
    ENDIF
ENDTRANSACTION
```

Return Value

Logical

Note

ROLLBACK() returns .T. if successful, or false (.F.) if there is no open transaction or the connection to the backend fails.

See Also

```
BEGIN TRANSACTION Command
COMMIT() Function
ENDTRANSACTION Command
OPEN DATABASE Command
```

ROUND()

ROUND(nExpression, nDecimals)

Remarks

Rounds a numeric expression to a specified number of decimal places.

Parameters

nExpression

Numeric expression to round.

nDecimals

Numeric expression indicating number of decimal places.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

SET DECIMALS TO 4
SET FIXED ON
```

```
? ROUND(123.4567,3) &&
? ROUND(123.4567,2) &&
? ROUND(123.4567,0) &&
```

Return Value

Numeric

See Also

[CEILING\(\)](#)

[FLOOR\(\)](#)

[INT\(\)](#)

[SET DECIM](#)

[SET FIXED](#)

RTOD()

RTOD(nExpression)

Remarks

Changes radians to degrees.

Parameters

nExpression

Numeric expression containing radians value.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? RTOD(ACOS(0))      && Displays 90.00
? RTOD(0.7854)        && Displays 45.00
```

Return Value

Numeric

Note

Often times used in conjunction with SIN(), COS(), TAN(), and DTOR().

See Also

COS() Function
DTOR() Function
SIN() Function
TAN() Function

RTRIM()

RTRIM(cExpression)

Remarks

Remove the blank spaces from the right side of the expression.

Parameters

cExpression

Character expression to use in RTRIM()

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? LEN(RTRIM( "HELLO WORLD!" ) )
```

Return Value

Character

See Also

[ALLTRIM\(\) Function](#)
[LTRIM\(\) Function](#)
[STR\(\) Function](#)
[TRIM\(\) Function](#)

SAVEPICTURE()

```
SAVEPICTURE(oPicture, cFileName [,nType])
```

Remarks

Saves a picture from a JAXBase picture class object.

Parameters

oPicture

JAXBase picture class object.

cFileName

Character expression holding a valid filename.

nType

Indicates in what format to save the picture base

Value	Picture Type
0 or omitted	BMP
1	JPG
2	PNG

Return Value

Logical

Note

SAVEPICTURE() returns TRUE if the operation is completed successfully, otherwise an error is generated if the operation fails.

See Also

LOADPICTURE()

SET SAFETY()

SEC()

SEC(tExpression)

Remarks

Returns the seconds portion from a DateTime expression.

Parameters

tExpression

Expression holding a DateTime value

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? SEC(DATETIME())
```

Return Value

Numeric

Note

SEC() will accept a Date

See Also

DATETIME()
HOUR() Function
MINUTE() Function
SET SECOND()

SECONDS()

`SECONDS([tExpression | cTime])`

Remarks

Returns the number of seconds since midnight.

Parameters

tExpression

Expression that contains a DateTime Value

cTime

Character expression that contains a Time value in 12 or 24 hour format.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? SECONDS()
? SECONDS({01/31/2025 01:00:45}) && 45
? SECONDS("12:14:00 PM")
? SECONDS("12:14 PM")
? SECONDS("13:45:18")
```

Return Value

Numeric

Note

If parameter **tExpression** is provided, returns the number of seconds since midnight. If a valid parameter is provided, returns the number of seconds since midnight represented by that value. SECONDS() will accept a DateTime value and return 0.

See Also

[DATE\(\)](#)
[DATETIME\(\)](#)
[HOURS\(\)](#)
[MINUTES\(\)](#)
[TIME\(\)](#)
 Seconds Property

SEEK()

```
SEEK(eExpression [, nWorkArea | cAlias [, nIndex | cIndexName]])
```

Remarks

Searches an indexed table for the first match of a record whose index key matches the provided eExpression.

Parameters

eExpression

Expression that holds the value to search for in the index.

nWorkArea

Numeric expression indicating the work area to

cTableAlias

Character expression indicating the alias of the

nIndex

Index number from the open index list.

cIndexName

Index name from the op

Example

* Demonstrate

CLEAR ALL

CLOSE ALL

SET EXCLUSI

SET CONSOLI

ACTIVATE CO

CLEAR

USE (_JAXFO

SET INDEX T

* The following example shows how to search for a record
* pointer to state and county
* record in state
? SEEK("YONKERS")
? RECNO(),state,county

Return Value

Logical

Note

You can only SEEK() on a table that has an open index.

eExpression must be the same type as the key expression in the searched index.

If SET NEAR is ON then if SEEK() fails to find the correct record, the next closest value is selected, otherwise the record pointer is placed after the end of file.

See Also

[EOF\(\) Function](#)
[FOUND\(\) Function](#)
[INDEXSEEK\(\) Function](#)
[LOCATE Command](#)
[Seek Command](#)
[SET NEAR Command](#)
[SET KEY Command](#)

SELECT()

```
SELECT ([0|1|2|3|cTableAlias])
```

Remarks

Returns the number of the currently selected work area, the highest unused under 36768, the lowest unused workarea, the next highest unused workarea over 36767, or the workarea of the indicated alias name.

Parameters

- 0 – Returns the number of the current workarea
- 1 – Returns the number of the highest unused work area
- 2 – Returns the number of the lowest unused workarea
- 3 – Returns the number of the next highest unused workarea

cTableAlias – character expression of table alias

Example

```
* Demonstrate SELECT()
CLEAR ALL
CLOSE ALL
SET COMPATIBLE OFF  &&
USE Company
? "Test 1=",SELECT(0)
? "Test 2=",SELECT(1)
? "Test 3=",SELECT(2)
? "Test 4=",SELECT(3)
? "Test 5=",SELECT("C
? "Test 6="
```

Return Value

Numeric

Note

SELECT() returns the number of the current work and is the same as SELECT(0).

An uninitialized variable is set to zero when a SELECT command or SELECT() function references it.

See Also

[ALIAS\(\) Function](#)
[SELECT Command](#)

SET()

```
SET(cSetting [, 1|2|3|4])
```

Parameters

cSetting

Specifies a character expression of the SET command for which you want to return information. The current setting of the specified command is returned as a character or numeric string.

1|2|3|4

Specifies additional information to return about a SET command.

Setting	Description	Status
ALTERNATE	Returns the alternate output file specification.	
ALTERNATE,1	Returns the alternate output file name.	
ASSERTS		
AUTOINCERROR		
AUTOSAVE		
BELL,1		ON or OFF
BLOCKSIZE		nBlockSize
CARRY		ON or OFF
CENTURY	Returns if the century is included in dates.	ON or OFF (default is ON)
CENTURY,1		nCentury (default is 19)
CENTURY,2		nRollOverYear (default is 75)
CLASSLIB	Returns the class library name.	cClassLibName
COLLATE		ON or OFF
CONFIRM		ON or OFF
CONSOLE		ON or OFF
COVERAGE		ON or OFF
COVERAGE		cFileName
CPCOMPILER		nCodePage
CPDIALOG		ON or OFF
CURRENCY		LEFT or RIGHT
CURRENCY		cCurrencySymbol
CURSOR		ON or OFF
DATABASE		cDataBaseName
DATASESSION		nDataSession
DATE		AMERICAN, ANSI, BRITISH, FRENCH, GERMAN, ITALIAN, JAPAN, USA, MDY, DMY or YMD
DATE,1		Date Order: 0 – MDY1-MDY2-YMD
DEBUG		ON or OFF
DEBUGOUT		cFileName
DECIMALS		nDecimals
DEFAULT	Current default folder. JAXBase starts in the JAXBase folder found in Documents.	cDirectory
DEFINITION	Provides a list of definition files recognized by the system, separated by semi colons.	cFileName

DELETED		ON or OFF
DELIMITERS		ON or OFF
DELIMITERS,1		cDelimiters
DEVELOPMENT		ON or OFF
DEVICE		SCREEN, PRINTER, or FILE
DEVICE,1		cDeviceName
ESCAPE		ON or OFF
EVENTLIST		Comma delimited list of events
EVENTTRACKING		ON or OFF
EVENTTRACKING,1		cFileName
EXACT		ON or OFF
EXCLUSIVE		
FDOW		
FIELDS		
FIELDS,1		, cFieldName2,...
FIELDS,2		OBAL
FILTER		tion
FIXED		
FULLPATH		
FWEEK		er
HEADINGS		
HELP		ON or OFF
HELP,1		cFileName
HELP,2		cCollectionURL
HELP,3		System
HOURS		12 or 24
INDEX		cIndexExpression
KEY		eExpression2, eExpression3
KEY,1		eExpression2
KEY,2		eExpression3
KEYCOMP		DOS or WINDOWS
LIBRARY		cLibraryName
LOCK		ON or OFF
LOGERROR		ON or OFF
MACKEY		cKey
MARGIN		nMargin
MEMOWIDT		nWidth (default is 0)
MESSAGE		nRow
MESSAGE,1		cMessageText
MOUSE		ON or OFF
MOUSE,1		nSensitivity
MULTILOCKS		ON or OFF
NEAR		ON or OFF
NOCPTRANS		comma delimited list of fields
NOTIFY		ON or OFF
NOTIFY,1		ON or OFF
NULL		ON or OFF
NULLDISPLAY		cDisplayString (default is .NULL.)
ODOMETER		nNumberOfRecords (default is 100)

ORDER		TAG cTagName OF cCDXFileName, cIDXFileName or blank
PALLETTE		ON or OFF
PATH		cPath
POINT		cDecimalPtChar (default is ".")
PRINTER		ON or OFF
PRINTER,1		cFileName or cPortName
PRINTER,2		Default Windows printer name
PRINTER,3		Default JAXBase printer name
PROCEDURE		cPathAndName
READBORDER		ON or OFF
REFRESH		nSeconds1
REFRESH,1		nSeconds2
REPROCESS		Current session setting
REPROCESS,1		System session setting
REPROCESS,2		Current session setting type 0 is returned if REPROCESS is set to attempts. 1 is returned if REPROCESS is set to seconds.
REPROCESS,3		System session setting type 0 is returned if REPROCESS is set to attempts. 1 is returned if REPROCESS is set to seconds.
RESOURCE		ON or OFF
RESOURCE,1		cFileName
SAFETY		ON or OFF
SECONDS		ON or OFF
SEPARATOR		cSeparatorChar
SPACE		ON or OFF
SQLBUFFERING		ON or OFF
STATUS		ON or OFF
STATUSBAR		ON or OFF
STRICTDATE		ON or OFF
SYSFORMATS		ON or OFF
SYSCMENU		ON, OFF, or AUTOMATIC
TABLEPROMPT		ON or OFF
TABLEVALIDATE		nLevel
TALK		ON or OFF
TALK,1		WINDOW, NOWINDOW or cWindowName
TEMPPFOLDER	Folder where temp files are written by JAXBase	
TEXTMERGE		ON or OFF
TEXTMERGE,1		cLeftDelimiter and cRightDelimiter
TEXTMERGE,2		cFileName
TEXTMERGE,3		SHOW or NOSHOW
TEXTMERGE,4		Evaluate source of TEXT...ENDTEXT call and return nesting level

TOPIC		cHelpTopicName Expression
TOPIC,1		nContextID
UDFPARMS		VALUE or REFERENCE
UNIQUE		ON or OFF

[Return Value](#)

[JAXBase data type](#)

[See Also](#)

[DISPLAY STATUS Command](#)

[LIST Commands](#)

[SET Command Overview](#)

DRAFT

SETFLDSTATE()

Status

Slated for Version 2.0

DRAFT

SIGN()

SIGN(nExpression)

Remarks

Returns -1, 0, or 1 depending on if the number is negative, zero, or positive.

Parameters

nExpression

Numeric expression to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? SIGN(PI())  && Displays 1
? SIGN(-423)  && Displays -1
```

Return Value

Numeric

See Also

[ABS\(\) Function](#)
[COS\(\) Function](#)
[SIN\(\) Function](#)

SIN()

SIN(nExpression)

Remarks

Convert degrees (expressed in radians) to the sine value.

Parameters

nExpression

Numeric expression holding degree value expressed as radians

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? SIN(DTOR(45))      && Displays 0.71
```

Return Value

Numeric

See Also

[ACOS\(\) Function](#)
[COS\(\) Function](#)
[DTOR\(\) Function](#)
[RTOD\(\) Function](#)
[TAN\(\) Function](#)
[SET DECIMALS Command](#)

SOUNDEX()

SOUNDEX(cExpression)

Remarks

Returns a phonetic representation of the specified character expression.

Parameters

cExpression

Character expression to use in the soundex calculation.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

```
? SOUNDEX( "JAXBase" )  
? SOUNDEX( "Allen" )
```

Return Value

Character

Note

The formula for creating a soundex representation is taken from

<https://physics.nist.gov/cuu/Reference/soundex.html>

SOUNDEX() returns a four-character string and is not case sensitive.

See Also

DIFFERENCE() Function

SPACE()

SPACE(nExpression)

Remarks

Returns a string containing nExpression number of spaces.

Parameters

nExpression

Numeric expression specifying how many spaces to return.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? SPACE(10)+"HELLO!"
? REPLICATE("-+",8)
```

Return Value

Character

See Also

PADC() | PADL() | PADR() Functions

REPLICATE() Function

SQRT()

SQRT(nExpression)

Remarks

Returns the square root of a number.

Parameters

nExpression

Numeric value from which to get the square root.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? SQRT(4)      && Returns 2
? SQRT(144)    && Returns 12
? SQRT(42)     && Returns "The Answer to Life, the Universe, and Everything" (kidding) 6.48
```

Return Value

Numeric

Note

The number of decimal places returned is subject to SET DECIMALS.

See Also

COS() Function

INT() Function

SIN() Function

TAN() Function

SET DECIMALS Command

STR()

`STR(nExpression [,nLength [,nDecimals]])`

Remarks

Returns a character expression from a numeric with formatting options.

Parameters

nExpression

Numeric expression to convert into a string.

nLength

Numeric expression indicating the length of the string.

nDecimals

Numeric expression indicating the number of decimal places.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? STR(10,6,2)      && Returns " 10.00"
? STR(99998473,6,2) && Returns "****.***"
```

Return Value

Character

Note

If the number will not fit into the space indicated with nLength, then an invalid number character string will appear with all asterisks.

STR() will round the value displayed if the number of decimals in nExpression exceeds nDecimals.

See Also

[VAL\(\) Function](#)

[STRCONV\(\) Function](#)

STRCONV()

Status

Slated for Version 2

DRAFT

STREXTRACT()

```
STREXTRACT(cSearchExpression, cBeginDelim [, cEndDelim [, nOccurrence [, nFlag]]])
```

Remarks

Extracts a string bounded by two delimiters.

Parameters

cSearchExpression

Character expression used to extract information bounded by delimiters.

cBeginDelim

Character expression holding the left delimiter.

cEndDelim

Character expression holding the right delimiter.

nOccurrence

Numeric expression indicating which occurrence to retrieve.

nFlag

Numeric expression indicating options on the retrieval. The values are additive.

Value	Description
1	Case-insensitive search.
2	End delimiter optional. If the end delimiter is not found, the rest of the string is returned.
4	Include the delimiters in the expression.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

lcStr="Title: Bourne Supremacy Length: 108m Gross: $291M"
? STREXTRACT(lcStr,"Title: "," Length:")           && Returns "Bourne Supremacy"
? STREXTRACT(lcStr,"Length: "," Gross:")          && Returns "108m"
? STREXTRACT(lcStr,"gross: "," ",1,5)             && Returns "$291M"

lcStr="123|144|155"
? STREXTRACT(lcStr,"| ","| ",2,2)                 && Returns "155"
```

Return Value

Character

Note

The default is to perform a case sensitive search, with both delimiters required (nFlag omitted or 0).

If cBeginDelim is empty, then the search starts at the beginning of the string. If cEndDelim is empty, the search ends at the end of the string.

See Also

[ATOKENS\(\) Function](#)

[STRTRAN\(\) Function](#)

[AT\(\) Function](#)

[RAT\(\) Function](#)

[SUBSTR\(\) Function](#)

STRFORMAT()

`STRFORMAT(cExpression [, eExpression1 [, eExpression2 ...]])`

Status

Slated for Version 1.0

Remarks

Merges values into a string formatted use specifier codes.

Parameters

cExpression

Character expression containing codes that indicate how to merge the eExpressions into the string.

eExpression1, eExpression2 ...

Expressions that will be merged into cExpression.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

* The following returns "Dear John, you own me $12.43"
? STRFORMAT("Dear {0}, you owe me {0:c}", "John", 12.43)

? STRFORMAT("0:0.000", 523.99437)      && Returns "523.994"

* Returns "one for the money and 2 for the show"
? STRFORMAT("{%1} for the money and {%2} for the show", "one", 2)

? STRFORMAT("{0} + {0} = {1}", 1, 2)      && Returns "1 + 1 = 2"
```

Return Value

Character

Note

STRFORMAT() uses codes like some popular languages where each {code} token accepts a value from the eExpression list. If there are 7 {code} tokens in the string, 7 eExpressions are expected. If there are not the right amount, an error is generated.

String format data {x} simply requires that you have the same number of eExpressions as the different {x} codes. Thus, if you have {0}, {1}, and {2} but the first two are repeated several times, it only requires you to have three eExpressions to convert.

The following table shows various codes to use in the {} brackets.

When you see x:, that stands for a number starting at zero and increasing with each {code} token.
Such as the following example: "{0:c}, {1:c}, {2}"

Number Formats

Code	Data Type	Description
{x:c}		
{x:e}		
{x:f}		
{x:g}		
{x:n}		

Custom Number Formats

Code	Data Type	Description
{x:00.00}		
{x:(0).##}		
{x:0.000}		
{x:0,0}		
{x:0,0%}		

Date and Time Formats

Code	Data Type	Description
{x:d}		
{x:D}		
{x:t}		
{x:T}		
{x:f} or {x:F}		
{x:g} or {x:G}		
{x:M}		
{x:r}		
{x:s}		
{x:u}		
{x:U}		
{x:Y}		

String Format

Code	Data Type	Description
{x}		
{x,10}		
{x,-10}		

See Also

[STR\(\)](#)
[TOSTRING\(\)](#)
[TRANSFORM\(\)](#)

DRAFT

STRTOFILE()

STRTOFILE(cExpression, cFileName [,nFlag])

Remarks

Write a file or append a string to a file.

Parameters

cExpression

Character expression holding a string to write to the file.

cFileName

Character expression holding file name to which the string is written.

nFlag

Numeric expression indicating how to write the data to the file.

The default is 0 for Overwrite with no terminator. Values are additive except for 4 & 8 which are mutually exclusive and will generate an error if both bits are set.

Value	Bit	Description
1	0001	Overwrite if 0, while 1 indicates to append to the file.
2	0010	Add text terminator if bit is set. Windows: character bytes 0x0A and 0x0D (10 and 13) Linux: character byte 0x0D (13)
4	0100	Unicode support slated for Version 2
8	1000	Unicode support slated for Version 2

Return Value

Numeric

Note

The return value is how many bytes were written to the file.

See Also

[FILETOSTR\(\) Function](#)

[SET SAFETY Command](#)

[FILE Class](#)

STRTRAN()

`STRTRAN(cExpression, cSearchFor, cReplaceWith)`

Remarks

Replace one or more occurrences of a string found inside cExpression.

Parameters

cExpression

The character string expression to search.

cSearchFor

The character expression to search for.

cReplaceWith

The character expression to replace matches of cSearchFor.

nStartOccurrence

Numeric expression indicating which occurrence to start with.

nNumberToReplace

Numeric expression indicating how many occurrences to replace.

nFlags

nFlags	Description
0 or omitted	Search is case-sensitive, and replacement is performed with exact cReplacement text (default).
1	Search is case-insensitive, and replacement is performed with exact cReplacement text.
2	Search is case-sensitive, and replacement is performed with cReplacement altered to match the case of the found.
3	Search is case-insensitive, and replacement is performed with cReplacement altered to match the case of the found.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
lcStr="The quick brown fox jumped over the lazy dog"
? STRTRAN(lcStr,"T","t",-1,-1,1)
? STRTRAN(lcStr,"dog","cat")
? STRTRAN(lcStr," ","_",3,2,0)
```

Return Value

Character

Note

Omitting everything after cReplaceWith or using -1 for nStartOccurrence and nNumberToReplace will cause all occurrences to be replaced.

See Also

[ATOKENS\(\) Function](#)

[CHRTRAN\(\) Function](#)

[STUFF\(\) Function](#)

DRAFT

STUFF()

`STUFF(cExpression, nStart, nLength, cReplace)`

Remarks

Creates a character string by replacing a number of characters in an expression with another expression.

Parameters

cExpression

Original character string that will be modified.

nStart

Numeric expression indicating which character to start the replacement.

nLength

Numeric expression indicating how many characters in the original string will be removed starting at the tnStart character.

cReplace

String to insert in that location.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Return Value

Character

Note

If nStart and nLength are greater than 0, then starting at the character indicated by nStart, that character and for a total of nLength, those characters will be removed and cReplace put in that location instead.

If nStart is greater than the length of cExpression, cReplace will be appended to cExpression.

If nStart is 0, then cReplace will be placed before the first character of cExpression and nStart will be ignored.

If nLength is 0, then cReplace will be placed before the character indicated by nStart and inserted. None of the cExpression characters will be removed.

If nStart or nLength are negative, an error will be generated.

See Also

[LEFT\(\) Function](#)

[PADC\(\) | PADL\(\) | PADR\(\) Functions](#)

[RIGHT\(\) Function](#)

[STREXTRACT\(\) Function](#)

[STRTRAN\(\) Function](#)

[SUBSTR\(\) Function](#)

STUFFC()

Status

Slated for Version 2.0

DRAFT

SUBSTR()

`SUBSTR(cExpression, nStart [, nLength])`

Remarks

Returns a character string by copying characters starting at nStart for nLength characters.

Parameters

cExpression

Expression to extract the sub-string from.

nStart

Numeric expression containing the starting position of the character to start the copy.

nLength

Numeric expression containing the number of characters to copy.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? SUBSTR("LeftSide",1,4)
? SUBSTR("Know your rights",11,5)
```

Return Value

Character

Note

If nLength is greater than the number of characters available after nStart, then only the remaining characters will be copied. If nStart is past the end of cExpression, an empty string will be returned; the same as if nLength is 0.

If nStart or nLength are negative, an error is generated.

See Also

[LEFT\(\) Function](#)

[RIGHT\(\) Function](#)

[STREXTRACT\(\) Function](#)

[STUFF\(\) Function](#)

SUBSTRC()

Status

Slated for Version 2.0

DRAFT

TAN()

TAN(nExpression)

Remarks

Returns the tangent of an angle.

Parameters

Returns the tangent of an angle expressed in radians.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? TAN(DTOR(0)))      && Returns 0.00
? TAN(DTOR(45))      && Returns 1.00
? TAN(DTOR(90))      && Generates meaningless large number
```

Return Value

Numeric

See Also

[COS\(\) Function](#)
[DTOR\(\) Function](#)
[RTOD\(\) Function](#)
[SET DECIMALS Command](#)
[SINE\(\) Function](#)

TARGET()

Status

Slated for Version 2

DRAFT

TEXTMERGE()

Remarks

Parameters

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Return Value

Note

See Also

TIME()

`TIME([nExpression | tExpression])`

Remarks

Returns the current time or converts a value to the current time.

Parameters

nExpression

Numeric expression containing a value of seconds to convert to a Time string.

tExpression

DateTime expression which will have it's Time component converted.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET HOURS TO 12
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? TIME(60)
? DATETIME(), TIME(DATETIME())
```

Return Value

Character

Note

If the value of nExpression is less than 0 then an empty character value is returned. If the value is greater than 86399 then the nExpression % 86400 is used to return the time. Any digits to the right of the decimal point are rounded off to milliseconds (3 decimal places).

The TIME() command is affected by the SET HOURS command.

If no parameters are included, TIME() returns the current system time from the operating system.

See Also

% Operator

DATETIME() Function

MOD Command

SET HOURS Command

TOSEC() Function

TRANSFORM()

Remarks

Parameters

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Return Value

Note

See Also

TRIM()

`TRIM(cExpression)`

Remarks

Remove the blank spaces from the right side of the expression.

Parameters

cExpression

Character expression to use in TRIM()

Example

```
* Demonstrate TRIM()
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? LEN( TRIM( "HELLO WORLD!           " ) )
```

Return Value

Character

See Also

[ALLTRIM\(\) Function](#)
[LTRIM\(\) Function](#)
[STR\(\) Function](#)
[RTRIM\(\) Function](#)

TTOC()

TTOC(tDateTime [, nFlag])

Remarks

Converts the given DateTime value to a formatted string.

Parameters

tDateTime

Expression holding the DateTime value to convert.

nFlag

Numeric expression indicating format.

Value	Description
0 or omitted	Returns the expression in the current regional format. SET CENTURY, SET HOUR, and SET SECONDS affect the result.
1	Returns expression suitable for indexing in the format yyyyymmddhhmmss. Hours are expressed in military time. SET CENTURY, SET HOUR, SET SECONDS settings do not affect the result.
2	Returns the time expression in the format hh:mm:ss or hh:mm:ss AM/PM SET SECONDS will affect if the :ss portion is returned. SET HOUR will affect if it returns military time or standard AM/PM time.
3	Returns the DateTime expression in the format yyyy-mm-ddThh:mm:ss which is useful when setting a date in a SQL backend database. Hours are expressed in military time. SET CENTURY, SET HOUR, SET SECONDS settings do not affect the result.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? TTOC(DATETIME())
? TTOC(DATETIME(),1)
? TTOC(DATETIME(),2)
? TTOC(DATETIME(),3)
```

Return Value

Character

Note

An invalid parameter sent as either parameter returns an empty value.

TTOC() will also accept a Date value, but the time is always returned as midnight (00:00:00).

See Also

DATE() Function
DATETIME() Function
HOUR() Function
MINUTE() Function
SEC() Function
SECOND() Function
SET HOURS Command
SET DATE Command
SET SECONDS Command
TIME() Function
TOSEC() Function
TTOD() Function

TTOD()

TTOD(tDateTime)

Remarks

Returns a DateTime value as a Date value.

Parameters

tDateTime

Expression containing the DateTime value to convert.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

a=DATETIME()
? a
? TTOD(a)
```

Return Value

Date

See Also

- DATE() Function
- DATETIME() Function
- HOUR() Function
- MINUTE() Function
- SEC() Function
- SECONDS() Function
- SET CENTURY Command
- SET HOUR Command
- SET SECOND Command
- TIME() Function
- TOSEC() Function
- TTOC() Function

TOSEC()

`TOSEC(cTime | tTime)`

Remarks

Returns a numeric value indicating number of seconds since midnight for the specified value.

Parameters

cTime

Expression containing the time value as a string in either 12 or 24 hour format.

tTime

Expression containing the DateTime value to convert.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
ACTIVATE CONSOLE
CLEAR

? TOSEC(DATETIME())
? TTOD("01:14:15 AM")
? TTOD("13:14:15")
? TTOD("01:14")
? TTOD("13:14")
```

Return Value

Date

See Also

[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[HOUR\(\) Function](#)
[MINUTE\(\) Function](#)
[SEC\(\) Function](#)
[SECONDS\(\) Function](#)
[SET CENTURY Command](#)
[SET HOUR Command](#)
[SET SECOND Command](#)
[TIME\(\) Function](#)
[TTOC\(\) Function](#)

TXNLEVEL()

Status

Slated for Version 2.0

DRAFT

TXTWIDTH()

```
TXTWIDTH(cExpression, cFont, nSize [, cStyle])
```

Remarks

Returns the width of a string in pixels based on the current or specified font information.

Parameters

cExpression

Expression containing the character string to measure.

cFont

Required character expression holding the name of the font to use. The font name will match the first available font in a case-insensitive search. In Windows, "COURIER" will usually match "Courier New" while in Linux it will usually match "Courier".

nSize

Required numeric expression indicating font size.

cStyle

Optional character expression indicating font styles. Omitted or empty string means normal style.

Character	Font Style
B	Bold
I	Italic
O	Outline
Q	Opaque
S	Shadow
-	Strikeout
T	Transparent
U	Underline

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? TXTWIDTH( "hello!", "Courier", 10)
```

Return Value

Numeric

Note

If the font and size combination is not available, TXTWIDTH will return 0.

See Also

[GETFONT\(\) Function](#)

[LEN\(\) Function](#)

DRAFT

TYPE()

TYPE(cExpression)

Remarks

Returns the type of the expression.

Parameters

cExpression

Character expression holding the expression whose type will be returned.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

DIMENSION aTest[2]
s="HELLO"
a = 4
b = 2
n=.NULL.

? TYPE('s')           && Returns C
? TYPE('a')           && Returns N
? TYPE('a>b')         && Returns L
? TYPE('aTest')        && Returns A
? TYPE('aTest[1]')     && Returns L
? TYPE('Z')            && Returns U
? TYPE('n')            && Returns X
? TYPE('DATE()')       && Returns D
? TYPE('{1/2/1996}')   && Returns D
? TYPE('4+3')          && Returns N
```

Return Value

Characer

Note

If you pass an array without element information, the value returned is "A". To get the value of an array element, provide the element information.

cExpression must contain a valid expression. In the above example, TYPE(s) would be the same as TYPE('HELLO') and would generate a variable not found error.

TYPE() should not be used to evaluate UDFs (user defined functions) as an error can be generated by the UDF if an incorrect parameter is passed.

The possible values returned are as follows.

Value	Description
A	Array – must include element information to get type
C	Character, Varchar, Varchar (binary)
D	Date
G	General
L	Logical
M	Memo field
N	Numeric, Float, Double, or Integer
O	Object
Q	Varbinary field
T	DateTime
U	Unidentified
W	Blob field
X	Null
Y	Currency Field

See Also

- [DATE\(\) Function](#)
- [DATETIME\(\) Function](#)
- [EVALUATE\(\) Function](#)
- [VARTYPE\(\) Function](#)
- [ISNULL\(\) Function](#)

UNBINDEVENTS()

Status

Slated for Version 2.0

DRAFT

UPDATED()

Remarks

Parameters

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR
```

Return Value

Note

See Also

UPPER()

UPPER(cExpression)

Remarks

Changes the characters in cExpression to upper case

Parameters

cExpression

Character expression to change to upper case.

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET EXCLUSIVE ON  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? UPPER( "alter ego" )
```

Return Value

Charcter

See Also

[ISALPHA\(\) Function](#)
[ISLOWER\(\) Function](#)
[ISUPPER\(\) Function](#)
[LOWER\(\) Function](#)
[PROPER\(\) Function](#)

USED()

`USED([nWorkarea | cAlias])`

Remarks

Returns if a table is in use in the specified work area, or if the specified alias name is in use.

Parameters

nWorkArea

Numeric expression indicating the work area to check.

cTableAlias

Character expression indicating the alias of the table to check.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET EXCLUSIVE ON
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

USE (_JAXFolder+"data\counties") ALIAS JC

? USED()
? USED(select())
? USED('JC')
```

Return Value

Logical

Note

If no parameter is provided, then USED() uses the current work area.

If the nWorkarea is uninitialized or has no open table, USED() returns false (.F.).

If the cAlias value is not found, USED() returns false (.F.).

See Also

[ALIAS\(\) Function](#)
[SELECT\(\) Function](#)
[AUSED\(\) Function](#)

VAL()

VAL(cExpression)

Remarks

Convert a string into a numeric expression.

Parameters

cExpression

Character expression to convert into a numeric expression.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? VAL( "1" )
? VAL( "33.14" )
? VAL( "ABC" )
```

Return Value

Numeric

Note

If the character expression does not contain a numeric value, 0 is returned.

See Also

SET DECIMALS Command
STR() Function

VARTYPE()

VARTYPE(eExpression)

Remarks

Returns the data type of the expression

Parameters

eExpression

Expression for which a type is returned.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR
```

```
a=7
```

```
? VARTYPE( 1+2 )      && Returns N
? VARTYPE(a)          && Returns N
? VARTYPE( "a" )      && Returns C
```

Return Value

Character

Note

The possible values returned are as follows.

Value	Description
A	Array – must include element information to get type
C	Character, Varchar, Varchar (binary)
D	Date
G	General
L	Logical
M	Memo field
N	Numeric, Float, Double, or Integer
O	Object
Q	Varbinary field
T	DateTime
U	Unidentified
W	Blob field
X	Null
Y	Currency Field

See Also

[DATE\(\) Function](#)
[DATETIME\(\) Function](#)
[EVALUATE\(\) Function](#)
[VARTYPE\(\) Function](#)
[ISNULL\(\) Function](#)

DRAFT

VERSION()

VERSION(nExpression)

Status

Currently only nExpression 0 returns a valid value.

Remarks

Returns the JAXBase version in use.

Parameters

Numeric expression specifying what information is to be returned.

Example

```
* Demonstrate
CLEAR ALL
CLOSE ALL
SET CONSOLE ON
ACTIVATE CONSOLE
CLEAR

? VERSION()
? Version(1)
? Version(2)
```

Return Value

Numeric, Character

Note

Returns a value based on the following numeric expressions. Values greater than shown return an empty character value.

nExpression	Description
0 or omitted	JAXBase Version in M.mm format (Major.minor)
1	JAXBase Version Type: 0 – Runtime 1 – IDE
2	Current language setting: 0 – Version prior to Version 2.0

See Also

[GETENV\(\) Function](#)
[OS\(\) Function](#)

WEEK()

WEEK(dExpression | tExpression)

Remarks

Returns a numeric value indicating the week of the year.

Parameters

dExpression

Date expression to be used to return week number

tExpression

DateTime expression to be used to return week number

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? WEEK(DATE())
```

Return Value

Numeric

Note

WEEK() is affected by SET FDOW and SET FWEEK Command.

See Also

CDOW() Function
DAY() Function
DOW() Function
SET FDOW Command
SET FWEEK Command

XMLTOCURSOR()

XMLTOCURSOR(cExpression[, cCursor [,nFlags]])

Status

Slated for Version 2.0

Remarks

Converts an XML expression to a cursor and returns number of records created.

Parameters

cExpression

Expression holding XML text.

cCursor

Name of cursor to create.

nFlags

Specifies how to manage the XML text.

nFlag (Additive)	Description
1	Preserves white space in the data
2	Appends data to cCursor
4	Map character fields to Varchar if set, otherwise character fields
8	Ignore text data over 254 characters
16	Ignore binary fields over 254 characters

Return Value

Numeric

Note

If an error occurs, the created cursor may be lost and the appended cursor may be partially updated.

See Also

[CURSORTOXML\(\)](#)

[CURSORTOJSON\(\)](#)

[JSONTOCURSOR\(\)](#)

YEAR()

`YEAR(dExpression | tExpression)`

Remarks

Returns the year portion of a Date or DateTime expression.

Parameters

dExpression

Date expression to be used to return week number

tExpression

DateTime expression to be used to return week number

Example

```
* Demonstrate  
CLEAR ALL  
CLOSE ALL  
SET CONSOLE ON  
ACTIVATE CONSOLE  
CLEAR  
  
? YEAR(DATE())
```

Return Value

Numeric

Note

YEAR() returns the full year (1 – 9999) and is not affected by the SET CENTURY Command.

See Also

[DATE\(\) Function](#)

[DATETIME\(\) Function](#)

[SET CENTURY Command](#)

Notes

DRAFT

DRAFT

JAXBase Environment Objects and Variables

The JAXBase system has several objects and variables that provide information about the JAXBase environment, the operating system that it's running under, and the user interfaces. The following is a comprehensive description of each.



JAXBase Environment Variables

JAXBase environment variables are simple data types that hold a value related to the JAX environment. Typically, these variables hold something like a folder name or a value that reflects a setting in the application's configuration.

Most JAXBase environment variables cannot be altered and will generate an error if you attempt to alter them.

_JAXFolder

The system folder that contains the TOOLS, GRAPHICS, EXAMPLES, and other folders that are useful for developing solutions. This variable is read-only and is set up during installation. It can only be changed by running the installer again and installing the system folder to a new location.

_JAXHome

This points to the folder where the JAXBase system resides. Specifically, the IDE, runtime, and help files. These files should not be altered, but you may need to alter or append the configuration files stored here. This variable is read-only and set at startup.

_JAXTemp

This points to the JAXBase temporary folder, which on Windows is normally C:\PROGRAMDATA\JAXBase\Temp. This variable is read-only and can be altered by changing the STARTUP.JBI file. Care should be taken when altering JBI files as it can cause unexpected issues if something is set incorrectly.

JAXBase Environment Objects and Arrays

All system objects are assigned a unique 8-character session ID when created.

DRAFT

JAX

Property	Data Type	Description
BaseClass	Character	CUSTOM
Class	Character	_JAX
Name	Character	
Path	Character	Path where this application is located
Version	Numeric	Version is returned in Major.Minor format (ex: 2.14)
X64	Logical	.T. indicates the 64-bit version
Config	Character	Configuration file name
ProjectEditor		
TableEditor		
FileEditor		
ImageEditor		
ClassEditor		
FormEditor		
MenuEditor		
ReportEditor		
LabelEditor		

The _JAX object has the following exposed methods.

Method	Description
GETID	Returns an 8-character session ID in the format comprised of capital letters and numbers

CPU

The CPU object contains information regarding the computer running JAXBase. It is read-only. This object is initialized at startup and remains static for the duration of the session.

Property	Data Type	Description
BaseClass	Character	EMPTY
Class	Character	_JAX
CPUCount	Numeric	Number of CPUs reported by the OS
CoreCount	Numeric	Number of cores reported by the OS
ID	Character	Eight-character session ID
PhysicalRAM	Numeric	Amount of physical RAM reported by the OS
VirtualRAM	Numeric	Amount of virtual RAM reported by the OS
OSName	Character	Name of operating system (Windows/Linux)
OSVersion	Character	Version information
OSVMajor	Numeric	OS major version number
OSVMinor	Numeric	OS minor version number
OSDistro	Character	Windows: Pro, Home, Server, Data Center, etc. Linux (distro names): Ubuntu, Mageia, SUSE, etc.
TerminalType	Numeric	0 – Physical workstation or server console 1 – Windows Terminal Server or Linux RDP server 2 – VDI
TerminalDistro	Character	Empty unless Terminal Type is 2 Name of VDI provider

_DRIVES

Status

Full support slated for Version 2

_Drives is an object array that contains drive/volume/share information reported by the operating system. The array is updated automatically.

Property	Data Type	Description
AError	Array	Each element contains an _Error object
BaseClass	Character	EMPTY
Class	Character	_JAX
Free	Numeric	Remaining free megabytes
ID	Character	Eight-character session ID
Name	Character	Name of device, volume, or share
Online	Logical	Online
Parent	Character	Parent ID
Removeable	Logical	Removeable (network and removeable drives list as .T.)
Size	Numeric	Physical Size in megabytes – 1 if less than 1 MB in size
Type	Numeric	0 – Drive, 1 – Volume, 2 – Share

_KEYBOARD

Status

Full support slated for Version 1

The _KEYBOARD class is constantly updated by a background thread, providing information on the current keyboard status.

Property	Data Type	Description
BaseClass	Character	EMPTY
Class	Character	_JAX
ID	Character	Eight-character session ID
KeyCount	Numeric	Number of keys on the keyboard
FKeyCount	Numeric	Number of function keys (F1 – Fx)
LastKey	Numeric	Value of last key pressed or 0
CurrentKey	Numeric	Current key that is down (0 if none are currently pressed down)
Shift	Logical	.T. if Shift key is down
LeftAlt	Logical	.T. if Left Alt key is down
RightAlt	Logical	.T. if Right Alt key is down
Ctrl	Logical	.T. if Control key is down
Fn	Logical	.T. if Fn (function) key is held down
CapsLock	Logical	.T. if caps lock is set
InsMode	Logical	.T. if insert mode is set
NumLock	Logical	.T. if number lock mode is set (arrows do not work)
ScrollLock	Logical	.T. if scroll lock is set
Queue	Numeric	Length of queue
Next	Numeric	Next key at top of queue – remove using INKEY() or CLEAR command

_MONITORS

Status

Full support slated for Version 2.0

_MONITORS is an array of MONITOR objects which provide information on connected monitors reported by the operating system. Each monitor is represented by a separate element in the array.

Each element is updated when a change to the monitor properties it represents is detected.

Each MONITOR object is as follows and all properties are read-only.

Property	Data Type	Description
BaseClass	Character	EMPTY
Class	Character	JAX
ColorDepth	Numeric	Bits of color 1=monochrome, 4=16 colors, etc.
Height	Numeric	Number of pixel rows
ID	Character	Eight-character session ID
Order	Numeric	Based on the monitor virtual order chart below.
ScreenSaver	Logical	.T. if monitor appears to be in screen saver mode
Width		Number of pixel columns

Monitor Order Chart

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

_MOUSE

Status

Full support slated for Version 1.0

The _Mouse class is constantly updated so that you can get information quickly from one source.

Property	Data Type	Description
BaseClass	Character	EMPTY
Class	Character	_JAX
ID	Character	Eight-character session ID
LeftButton	Logical	.T. if down
MiddleButton	Logical	.T. if down
RightButton	Logical	.T. if down
Monitor	Numeric	Monitor region mouse is currently in (1 – 16). See _MONITOR.
MX	Numeric	Current pixel column for the current monitor
MY	Numeric	Current pixel row for the current monitor
X	Numeric	Current pixel column for the entire desktop region
Y	Numeric	Current pixel row for the entire desktop region

_PRINTERS

Status

Full support slated for Version 2.0

_Printers is an object array that holds information about all printers reported by the operating system.

Column	Data Type	Description
AError	Numeric	_ERROR array
BaseClass	Character	CUSTOM
Class	Character	_JAX
Connection	Character	Connection name, IP, or share
ConnectionType	Numeric	0 –Unknown, 1 – Local, 2 – Network Share, 3 – Client
ID	Character	Eight-character session ID
IsColor	Logical	.T. if a color printer
Location	Character	Physical location
Name	Character	Printer name
Manufacturer	Character	Printer manufacturer
Model	Character	Model number
Online	Logical	.T. if currently online
Tray	Array	Single element array of logical values indicating if paper is out. False (.F.) means the tray has paper or is in an unknown state.
Width	Numeric	Width in millimeters

Each _PRINTERS element has the following methods.

Method	Description
EjectPage	Eject a page
SendTest	Send a test page
GetQueue	Returns an array of jobs in the printer queue
ManageQueue	Send a queue command (Cancel, Clear, Restart)

_REGEX

Exposes the regular expression engine to JAXBase.

Property	Data Type	Description
AError	Array	Returns an array of _ERROR objects. Reset with each Regex call.
BaseClass	Character	CUSTOM
Class	Character	_JAX
ID	Character	8-character session ID
IgnoreCase	Logical	Directs Regex methods to ignore case if true (.T.)
Pattern	Character	Optional pattern to use for Regex calls
Text	Caracter	Optional text to use for Regex calls

The _REGEX object has the following methods.

Method	Description
GetMatches	<p>Returns an array of matches in the provided text, based on the Regex pattern GETMATCHES () Returns matches in the Text property using the Patterns property.</p> <p>GETMATCHES (cExpr) Returns matches in cExpr using the Patterns property.</p> <p>GETMATCHES (cText, cPattern) Returns matches in cText using patterns in cPattern</p>
Replace	<p>Returns the text with replacements from a string based on the Regex pattern REPLACE (cReplace) Returns Text with replacements of cReplace using the Patterns property.</p> <p>REPLACE (cText, cPattern, cReplace) Returns cText with replacement of cReplace using cPattern.</p>
Split	<p>Returns an array containing text being split based on the Regex pattern with optional case sensitivity criteria. SPLIT () Return an array created using the Text property being split using the Pattern and IgnoreCase properties.</p> <p>SPLIT (cText, cPattern) Return an array by splitting cText, using cPattern and IgnoreCase property.</p> <p>SPLIT (cText, cPattern, lIgnoreCase) Return an array by splitting cText, using cPattern and lIgnoreCase parameters.</p>

Note

Pattern and Text properties can come in handy when you wish to perform several different actions using the same pattern or text.

For more information, see [Using the Regular Expression Object](#).

DRAFT

DRAFT

JAXBase Class Reference

All user objects created in JAXBase are assigned, depending on the SET ID setting, an eight-character ID string which is unique to the session or a 36-character GUID() which is unique globally.

DRAFT

APPEND

Status

Basic support slated for Version 1.0

Full support slated for Version 2.0

DRAFT

BROWSE

Status

Basic support slated for Version 1.0

Full support slated for Version 2.0

Browse form used by the BROWSE command. You can have multiple active browse forms at one time.

Property	Data Type	Attributes	Default	Description
ActiveColumn	Numeric			
ActiveRow	Numeric			
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
AllowAddNew	Logical			
AllowAutoFit	Numeric			
Anchor	Numeric			
AllowCellSelection	Logical			
BackColor	Color			
BaseClass	Character	P	GRID	JAXBase class this control inherits from
Class	Character	P	BROWSER	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
ColumnCount	Numeric	P		
Columns	Collection			
DeleteMark	Logical			
Enabled	Logical			
FontBold	Logical			
FontItalic	Logical			
FontName	Character			
FontSize	Numeric			
FontStrikeThrough	Logical			
FontUnderline	Logical			
ForeColor	Color			
GridLineColor	Color			
GridLineWidth	Numeric			
GridLines	Numeric		3	0 – None, 1 – Horizontal, 2 – Vertical, 3 – Both
HeaderHeight	Numeric			
Height	Numeric			
Highlight	Logical			
HighlightBackColor	Color			
HighlightForeColor	Color			
ID	Character	P		JAXBase object ID assigned at initialization
Left	Numeric			
Name	Character			
Objects	Collection			
Parent	Object			
ParentClass	Character			
ReadOnly	Logical			

RecordMark	Logical			
RecordSource	Character			
RecordSourceType	Numeric			
RightToLeft	Logical			
RowColChange	Numeric			
RowHeight	Numeric			
ScrollBars	Numeric			
SelectedItemBackColor	Color			
SelectedItemForeColor	Color			
TabStop	Logical			
TabIndex	Numeric			
Tag	Character			Used to store extra information needed for the program. It is useful for storing a JSON string.
Top	Numeric			
Value				
Visible	Logical			
Width	Numeric			

Methods

Method	Description
AddColumn	
AddObject	
AddProperty	Adds a property to the control
AddMethod	Adds a user defined method to the control
DeleteColumn	
Refresh	
RemoveObject	
SetFocus	
ZOrder	

Events

Method	Description
AfterRowColChange	
BeforeRowColChange	
Click	
DblClick	
Deleted	
Destroy	Executed when class is released from memory
Error	If an unhandled error occurs, the error is written to the log and added to the AError array, if it is enabled, a JAXBase error will be generated.
Init	Executed when a class is created
KeyPress	
MiddleClick	
MouseDown	
MouseEnter	
MouseLeave	
MouseMove	
MouseUp	
MouseWheel	
Moved	
Resize	
RightClick	
Valid	
When	

Example

Notes

See Also

EDIT

Status

Basic support slated for Version 1.0

Full support slated for Version 2.0

DRAFT

_ERROR

The _ERROR object provides a unified error messaging system for returning error information. Many objects will have an AERROR array that provides information on all errors that occurred during the last transaction.

Properties

Property	Data Type	Attributes	Default	Description
BaseClass	Character	P	CUSTOM	JAXBase class this control inherits from
Class	Character	P	_ERROR	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
ErrorNo	Numeric	R	0	JAXBase Error Number
ID	Character	P		JAXBase object ID assigned at initialization
LineNo	Numeric	R	0	Line number where error occurred – 0 if not relevant
Message	Character	P		First message line from Messages
Messages	Character	R		Message text with all messages terminated with 0x0D
Procedure	Character	R		Procedure where error occurred
RefObject	Object	R	.NULL.	Pointer to object that is in error
RefObjectError	Numeric	R	0	Error number returned by driver (ODBC, network, etc)

Events

Method	Description
INIT	Sets the properties from the parameters passed in
ERROR	If an error occurs, the error is written to the log, if it is enabled, and a JAXBase error will be generated.

Example

```

* This is a model of how the ERROR event works in JAXBase objects
PROCEDURE Error
LPARAMETERS nErrNo, nErrLine, cMsg, cProcedure, oObj, nObjErrNo
LOCAL Idx

IF This.AError[1].ErrNo>0
  * Add to the error array
  Idx= alen(This.AError,2)+1
  DIMENSION This.AError[Idx]
ELSE
  * Fill in the first element
  Idx=1
ENDIF

* Create the _ERROR object and assign it to the array
This.AError[Idx]=CREATEOBJECT("_Error",nErrNo, nErrLine, cMsg, oObj, nObjErrNo, cProcedure)
ENDWITH

RETURN

```

Notes

You are encouraged to implement _ERROR object arrays when creating your custom error handlers to create a uniform method for error handling.

_ERROR object properties are assigned during initialization and cannot be altered after initialization.

The AERROR array will always have at least one element. If no error occurred, then AERROR[1].ErrorNo will equal zero.

See Also

Error() Function
Line() Function
Message() Function
ON ERROR Command
Procedure() Function
TRY/CATCH/FINALLY/ENDTRY Command
Error Method
Debugging and Error Handling

BARCODE

Status

Basic support slated for Version 1.0

Full support slated for Version 2.0

Properties

Property	Data Type	Attributes	Default	Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
BaseClass	Character	P	CUSTOM	JAXBase class this control inherits from
Class	Character	P	_ERROR	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
CalCheckSum	Logical		.T.	If the value is set to true (.T.), then any required checksum is automatically calculated and added to the text. PlainText must be set to true (.T.) before the checksum will be calculated.
CheckSum	Character			The WRITE method sets this to the calculated checksum if the check sum is calculated. The READ method extracts the check sum and places it into the CheckSum property. If no checksum was found, it is set to an empty string.
ErrorNo	Numeric	R	0	JAXBase Error Number (Default is 1525)
ID	Character	P		JAXBase object ID assigned at initialization
Image	Character			Encoded image file read by the Read method or created by the Write method.
PlainText	Logical		.T.	If value is true (.T.), any prefix and suffix information for the text required for the barcode is added.
Scale	Numeric			Values above 1 create a taller image while values less than 1 create a smaller image. If the resolution of the output device can be determined, a scale of 1 will produce an image approximately 1" tall, otherwise the resolution defaults to 72dpi and scales accordingly.
Symbology	Numeric			See Reading and Writing Barcodes
Tag	Character			Used to store extra information needed for the program. It is useful for storing a JSON string.
Text	Character			Barcode text to encode or decoded from image

Methods

Method	Description
Read	
Write	

Events

Method	Description
AfterRead	Called after a barcode image is decoded with no errors
AfterWrite	Called after a barcode image is created with no errors
INIT	Sets the properties from the parameters passed in
ERROR	If an error occurs, the error is written to the log, if it is enabled, and a JAXBase error will be generated.

Example

Notes

See Also

[GETJSON\(\) Function](#)
[GETJSON\(\) Function](#)
[Reading and Writing Barcodes](#)

CHECKBOX

Status

Slated for Version 0.6

DRAFT

COLLECTION

Status

DRAFT

COMBOBOX

Status

Slated for Version 0.6

DRAFT

COMMANDBUTTON

A command button provides a way to initiate an event by the user. When clicked, the CLICK event is executed.

Property	Data Type			Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
Anchor	Numeric		0	Defines how the edges of the control are bound to the edges of the container
Autosize	Logical		.F.	Specifies if the control can automatically resize to fit the text in the caption.
BackColor	Numeric		240,240,240	Integer value representing the RGB background color
BaseClass	Character	P	LABEL	JAXBase class this control inherits from
Caption	Character		BUTTONx	Hold the text that is displayed by this control
Class	Character	P	LABEL	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
Enabled	Logical		.T.	Indicates if the control can respond to user events
FontBold	Logical		.F.	.T. sets the control's font as bold
FontItalic	Logical		.F.	.T. sets the control's font as italics
FontName	Character		Ariel	Name of font in use
FontSize	Numeric		10	Size of font
FontStrikeThrough	Logical		.F.	.T. sets the control's font as strike-through
FontUnderline	Logical		.F.	.T. sets the control's font as <u>underline</u>
ForeColor	Numeric		0,0,0	Integer value representing the RGB foreground color
Height	Numeric		150	Height, in pixels
ID	Character	P		JAXBase object ID assigned at initialization
Left	Numeric		0	Leftmost position of control in the parent
Name	Character	C	FORMx	Name of object
Parent	Object	R		Object to which the form is attached
ParentClass	Character	P		Class of parent object
RightToLeft	Logical		.F.	Displays text in right to left order
TabIndex	Numeric		0	This controls position in the tab order of controls
TabStop	Numeric		.T.	Indicates if this control can be reached via the TAB key
Top	Numeric		0	Topmost position of control in the parent
Visible	Logical		.T.	If .F. then this control is invisible
Width	Numeric		150	Width of control

C – Read/Write before initialization, read only once initialized

R – Read Only

P – Protected and cannot be modified

Methods

Method	Description
AddProperty	Adds a property to an object
AddMethod	Adds a user defined method to an object
MEUpdate	Allows you to insert new code into most methods or events before initialization. Methods or events that contain code after initialization cannot be updated.
Refresh	Repaints the control and updates any values

Events

Event	Description
Click	Executed when the user clicks on the control
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
GotFocus	Executed when a control receives focus
Init	Executed when a control is created
LostFocus	Executed when a control loses focus
MouseEnter	Executed when the user moves the mouse into the control
MouseLeave	Executed when the user moves the mouse out of the control
Valid	Executed before the control loses focus

Example

Notes

See Also

COMMANDGROUP

Status

Slated for Version 0.6

DRAFT

CONTAINER

Status

DRAFT

CUSTOM

The Custom class is a minimal class that has the following properties, events, and methods.

Property	Data Type	Attributes	Default Value	Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
BaseClass	Character	P	CUSTOM	JAXBase class this control inherits from
Class	Character	P	CUSTOM	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
Enabled	Logical		.T.	Indicates if the form can respond to user events
ID	Character	P		JAXBase object ID assigned at initialization
KeyPreview	Logical		.F.	.T. means the form will intercept keystrokes and send them to the KeyPress event
Left	Numeric		0	Leftmost position of form in the display area
Name	Character	D	FORMx	Name of form
Objects	Object[]	P		Array of child objects attached to form
Parent	Object	R		Object to which the form is attached
ParentClass	Character	P		Class of parent object

Methods

Method	Description
AddProperty	Adds a property to an object
AddMethod	Adds a user defined method to an object
AddObject	Adds an object to an object
MEUpdate	Allows you to insert new code into any method or event before initialization and any user defined method and event after initialization.
Refresh	Repaints the control and updates any values

Events

Event	Description
Click	Executed when the user clicks on the control
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
GotFocus	Executed when a control receives focus
Init	Executed when a control is created
LostFocus	Executed when a control loses focus
MouseEnter	Executed when the user moves the mouse into the control
MouseLeave	Executed when the user moves the mouse out of the control
Valid	Executed before the control loses focus

Example

Notes

See Also

DRAFT

EDITBOX

Status

Slated for Version 0.6

Property	Data Type			Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
Anchor		C		
BackColor	Numeric		240,240,240	Integer representing the RBG color of the background
BaseClass	Character	P	LABEL	JAXBase class this control inherits from
BorderStyle	Numeric		3	Border style: 0 – None 1 – Fixed Single 2 – Fixed Dialog 3 – Sizable (Default)
Class	Character	P	LABEL	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
Enabled	Logical		.T.	Indicates if the form can respond to user events
FontBold	Logical		.F.	.T. sets the controls font as bold
FontItalic	Logical		.F.	.T. sets the controls font as italics
FontName	Character		Ariel	Name of font to use
FontSize	Numeric		10	Size of font
FontStrikeThrough	Logical		.F.	.T. sets the controls font as <u>strikethrough</u>
FontUnderline	Logical		.F.	.T. sets the controls font as <u>underline</u>
ForeColor	Numeric		0,0,0	Color of font
Height	Numeric		150	Height, in pixels
ID	Character	P		JAXBase object ID assigned at initialization
Left	Numeric		0	Leftmost position of form in the display area
MaxLength	Numeric			Maximum length of text allowed in the control. 0 = no limit
Name	Character	C	FORMx	Name of form
Parent	Object	P		Object to which the form is attached
ParentClass	Character	P		Class of parent object
PasswordChar	Character			Mask character to obfuscate the display of the text
ReadOnly	Logical		.F.	When .T., the user cannot modify its contents
RightToLeft	Logical		.F.	Scale Factor between Text and Graphic user interfaces
ScrollBars	Numeric		2	0 – None, 1 – Horizontal, 2 – Vertical, 3 – Both
TabIndex	Numeric		0	
TabStop	Numeric		.T.	
Text	Character			
TextAlignment	Numeric			0 – Left, 1 – Right, 2 – Center
TextLength	Numeric		0	Length of text in the control
Top	Numeric		0	Topmost position of form in display area
Visible	Logical		.T.	
Width	Numeric		150	Width of form
WordWrap	Logical		.F.	Value of true (.T.) turns on wordwrap

C – Read/Write before initialization, read only once initialized

R – Read Only once initialized.

P – Protected and cannot be modified

Methods

Method	Protected	Description
AddProperty	Y	
AddMethod	Y	
AddObject	Y	
Destroy	C	
Init	C	
MEUpdate	Y	
RemoveProperty	Y	
RemoveObject	Y	

Event	Description
Click	Executed when the user clicks on the control
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
GotFocus	Executed when a control receives focus
Init	Executed when a control is created
LostFocus	Executed when a control loses focus
MouseEnter	Executed when the user moves the mouse into the control
MouseLeave	Executed when the user moves the mouse out of the control
Valid	Executed before the control loses focus

Example

Notes

See Also

EMPTY

The EMPTY class starts with no properties and does not support events or methods except for the ADDPROPERTY() and REMOVEPROPERTY() methods.

The EMPTY class is used as a basic object for storing information by various JAXBase commands.

See Also

GATHER NAME Command
SCATTER NAME Command

DRAFT

FILE

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

FORM

Remarks

All objects that are visible in JAXBase must be attached to a form class. A form provides the visible pallet area for displaying information and allowing user control.

Property	Data Type	Attributes	Default Value	Description
Activecontrol	Object	R		Direct reference to the current active control
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
AlwaysOnTop	Logical		.F.	Indicates if the form is always on top
AutoCenter	Logical		.F.	Is form centered on the viewing area when initialized
BackColor	Numeric		240,240,240	Integer representing the RBG color of the background
BaseClass			FORM	JAXBase Class this control inherits from
BorderStyle	Numeric		3	Border style: 0 – None, 1 – Fixed Single, 2 – Fixed Dialog, 3 – Sizable (Default)
Caption	Character		FORMx	Form's caption
Class	Character	P	FORM	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
Closable	Logical		.T.	If .T. then the close icon is visible and active
ControlCount	Numeric	R	0	Number of objects on the form.
DataSession	Numeric	D,1	1	Data session ID: 1 – Default, 2 – Private data session
Enabled	Logical		.T.	Indicates if the form can respond to user events
FontBold	Logical		.F.	.T. sets the controls font as bold
FontItalic	Logical		.F.	.T. sets the controls font as italics
FontName	Character		Ariel	Name of font to use
FontSize	Numeric		10	Size of font
FontStrikeThrough	Logical		.F.	.T. sets the controls font as <u>strikethrough</u>
FontUnderline	Logical		.F.	.T. sets the controls font as <u>underline</u>
ForeColor	Numeric		0,0,0	Color of font
Height	Numeric		150	Height, in pixels
Hwnd	Numeric	R	0	OS Handle of the form
Icon	Character			File name of icon placed in upper left corner
ID	Character	P		JAXBase object ID assigned at initialization
KeyPreview	Logical		.F.	Indicates if keystrokes are sent to the KeyPress event
Left	Numeric		0	Leftmost position of form in the display area
LockScreen	Logical		.F.	.T. changes to the form are hidden until set back to .F.
MaxButton	Logical		.T.	.T. means the maximize button is visible and enabled
MaxHeight	Numeric		-1	Maximum allowed height of form
MaxWidth	Numeric		-1	Maximum allowed width of form
MinButton	Logical		.T.	.T. means the minimize button is visible and enabled
MinHeight	Numeric		-1	Minimum allowed height of form
MinWidth	Numeric		-1	Minimum allowed width of form
Name	Character	D	FORMx	Name of form
Objects	Object[]	P		Array of child objects attached to form

Property	Data Type	Attributes	Default Value	Description
Parent	Object	R		Object to which the form is attached
ParentClass	Character	P		Class of parent object
ScaleFactor	Numeric	D,2	0	Scale Factor between Text and Graphic user interfaces
ScrollBars	Numeric		0	Type of scroll bars displayed on the form 0 – None (default), 1 – Horizontal, 2 – Vertical, 3 – Both
Top	Numeric		0	Topmost position of form in display area
Visible	Logical		.T.	Indicates if control is visible
Width	Numeric		150	Width of form

D – Read/Write before initialization, read only once initialized

R – Read Only

P – Protected and cannot be modified

1 – The data session is read/write before initialization, read only once initialized. Once initialized, the value in DataSession is the actual data session ID.

2 – Scale factors (Under Consideration)

Methods

Method	Description
AddProperty	Adds a property to an object
AddMethod	Adds a user defined method to an object
AddObject	Adds an object to an object
MEUpdate	Allows you to insert new code into any method or event before initialization and any user defined method and event after initialization.
Refresh	Repaints the control and updates any values

Event	Description
Click	Executed when the user clicks on the control
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
GotFocus	Executed when a control receives focus
Init	Executed when a control is created
LostFocus	Executed when a control loses focus
MouseEnter	Executed when the user moves the mouse into the control
MouseLeave	Executed when the user moves the mouse out of the control
Valid	Executed before the control loses focus

Example

Notes

While the Objects array cannot be modified, you can modify the properties of its elements.

See Also



FORMSET

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

FTP

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

GRID

Status

Slated for Version 0.8

DRAFT

HTTP

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

IMAGE

Status

Basic support slated for Version 0.6

Full support slated for Version 1.0



LABEL

Property	Data Type			Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
Anchor				
AutoSize				
BackColor	Numeric		240,240,240	Integer representing the RBG color of the background
BaseClass	Character	P	LABEL	JAXBase class this control inherits from
BorderStyle	Numeric		3	Border style: 0 – None 1 – Fixed Single 2 – Fixed Dialog 3 – Sizable (Default)
Caption	Character		LABELx	Form's caption
Class	Character	P	LABEL	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
Enabled	Logical		.T.	Indicates if the form can respond to user events
FontBold	Logical		.F.	.T. sets the controls font as bold
FontItalic	Logical		.F.	.T. sets the controls font as italics
FontName	Character		Ariel	Name of font to use
FontSize	Numeric		10	Size of font
FontStrikeThrough	Logical		.F.	.T. sets the controls font as strikethrough
FontUnderline	Logical		.F.	.T. sets the controls font as underline
ForeColor	Numeric		0,0,0	Color of font
Height	Numeric		150	Height, in pixels
ID	Character	P		JAXBase object ID assigned at initialization
Left	Numeric		0	Leftmost position of form in the display area
Name	Character	D	FORMx	Name of form
Parent	Object	R		Object to which the form is attached
ParentClass	Character	P		Class of parent object
RightToLeft	Logical		.F.	Scale Factor between Text and Graphic user interfaces
TabIndex	Numeric		0	
Top	Numeric		0	Topmost position of form in display area
Visible	Logical		.T.	
Width	Numeric		150	Wdith of form

D – Read/Write before initialization, read only once initialized

R – Read Only

P – Protected and cannot be modified

Methods

Method	Description
AddProperty	Adds a property to an object
AddMethod	Adds a user defined method to an object
MEUpdate	Allows you to insert new code into any method or event before initialization and any user defined method and event after initialization.
Refresh	Repaints the control and updates any values

Events

Event	Description
Click	Executed when the user clicks on the control
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
GotFocus	Executed when a control receives focus
Init	Executed when a control is created
LostFocus	Executed when a control loses focus
MouseEnter	Executed when the user moves the mouse into the control
MouseLeave	Executed when the user moves the mouse out of the control
Valid	Executed before the control loses focus

Example

Notes

See Also

LINE

Status

Slated for Version 0.6

DRAFT

LISTBOX

Status

Slated for Version 0.6

DRAFT

MENU

Status

Slated for Version 0.8

DRAFT

OPTIONBUTTON

Status

Slated for Version 0.6

DRAFT

OPTIONGROUP

Status

Slated for Version 0.6

DRAFT

PAGE

Status

Slated for Version 0.6

DRAFT

PAGEFRAME

Status

Slated for Version 0.6

DRAFT

PIPE

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

POP3

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

PRINTER

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

SHAPE

Status

Slated for Version 0.6

DRAFT

SMS

Status

Slated for after Version 2.0

DRAFT

SMTP

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

SOUND

Status

Basic support slated for Version 0.6

Full support slated for Version 2.0

Property	Data Type	Attributes	Default Value	Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
BaseClass	Character	P	CUSTOM	JAXBase class this control inherits from
Class	Character	P	CUSTOM	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
BitRate	Numeric			Gets or sets bit rate
Channels	Numeric			Gets or sets number of channels 1 – mono, 2 – stereo
Sound	Character			Name of sound file
Tag	Character			

Methods

Method	Description
AddProperty	Adds a property to an object
AddMethod	Adds a user defined method to an object
AddObject	Adds an object to an object
MEUpdate	Allows you to insert new code into any method or event before initialization and any user defined method and event after initialization.
Play	Plays the sound file
Refresh	Repaints the control and updates any values
Stream	Allows a sound to be played from a memory variable

Events

Event	Description
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
Init	Executed when a control is created
Loaded	Executed when after a sound file is loaded

Example

Notes

See Also

DRAFT

SPINNER

Status

Slated for Version 0.6

DRAFT

SQL

Status

Slated for Version 0.6

Property	Data Type	Attributes	Default Value	Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
BaseClass	Character	P	SOUND	JAXBase class this control inherits from
Class	Character	P	SOUND	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
ConnectionString	Character			
Database	Character	C		
Driver	Character	C		
Encrypted	Logical			
ID	Character	P		JAXBase object ID assigned at initialization
Instance	Character	C		
IsConnected	Logical	R	.F.	
Port	Numeric	C		
Server	Character	C		
UserID	Character	C		
UserPW	Character	C		

Protected

C – Conditional. Cannot be changed after while connected.

P – Protected. Cannot be changed.

Methods

Method	Description
AddProperty	Adds a property to an object
AddMethod	Adds a user defined method to an object
AddObject	Adds an object to an object
Connect	
Disconnect	
Exec	
MEUpdate	Allows you to insert new code into any method or event before initialization and any user defined method and event after initialization.
Refresh	Repaints the control and updates any values
Update	Pushes records from a local DBF into a SQL table

Events

Event	Description
Connected	
Destroy	Executed before a control is released from memory
Disconnected	
Error	Executed if an unhandled error occurs
Init	Executed when a control is created

Example

Notes

See Also

TEXTBOX

Property	Data Type			Description
AError	Array	P		Array of _ERROR class for errors captured by the ERROR event, if active.
Anchor		C		
BackColor	Numeric		240,240,240	Integer representing the RBG color of the background
BaseClass	Character	P	LABEL	JAXBase class this control inherits from
BorderStyle	Numeric		3	Border style: 0 – None 1 – Fixed Single 2 – Fixed Dialog 3 – Sizable (Default)
Class	Character	P	LABEL	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
Enabled	Logical		.T.	Indicates if the form can respond to user events
FontBold	Logical		.F.	.T. sets the controls font as bold
FontItalic	Logical		.F.	.T. sets the controls font as italics
FontName	Character		Ariel	Name of font to use
FontSize	Numeric		10	Size of font
FontStrikeThrough	Logical		.F.	.T. sets the controls font as <u>strikethrough</u>
FontUnderline	Logical		.F.	.T. sets the controls font as <u>underline</u>
ForeColor	Numeric		0,0,0	Color of font
Height	Numeric		150	Height, in pixels
ID	Character	P		JAXBase object ID assigned at initialization
Left	Numeric		0	Leftmost position of form in the display area
Name	Character	C	FORMx	Name of form
Parent	Object	P		Object to which the form is attached
ParentClass	Character	P		Class of parent object
PasswordChar	Character			Mask character to obfuscate the display of the text
ReadOnly	Logical		.F.	When .T., the user cannot modify its contents
RightToLeft	Logical		.F.	Scale Factor between Text and Graphic user interfaces
TabIndex	Numeric		0	
TabStop	Numeric		.T.	
Text				
Top	Numeric		0	Topmost position of form in display area
Visible	Logical		.T.	
Width	Numeric		150	Wdith of form

C – Read/Write before initialization, read only once initialized

R – Read Only once initialized.

P – Protected and cannot be modified

Methods

Method	Protected	Description
AddProperty	Y	
AddMethod	Y	
AddObject	Y	
Destroy	C	
Init	C	
MEUpdate	Y	
RemoveProperty	Y	
RemoveObject	Y	

Event	Description
Click	Executed when the user clicks on the control
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
GotFocus	Executed when a control receives focus
Init	Executed when a control is created
LostFocus	Executed when a control loses focus
MouseEnter	Executed when the user moves the mouse into the control
MouseLeave	Executed when the user moves the mouse out of the control
Valid	Executed before the control loses focus

Example

Notes

See Also

TCP

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

TIMER

Status

Slated for Version 1.0

DRAFT

UDP

Status

Basic Support by Version 1.0

Full support slated for after Version 1.0

DRAFT

VIDEO

Status

Basic support slated for Version 1.0

Full support slated for Version 2.0

Property	Data Type	Attributes	Default Value	Description
BaseClass	Character	P	VIDEO	JAXBase class this control inherits from
Class	Character	P	VIDEO	Name of the class control is based upon
ClassLibrary	Character	P		Indicates if the class is part of a class library
Codec	Character			
Duration	Numeric			Duration in seconds
FrameRate	Numeric			Number of frames per second
Height	Numeric			Name of sound file
Tag	Character			
Video	Character			Name of video file
Width	Numeric			

Methods

Method	Description
AddProperty	Adds a property to an object
AddMethod	Adds a user defined method to an object
AddObject	Adds an object to an object
MEUpdate	Allows you to insert new code into any method or event before initialization and any user defined method and event after initialization.
Pause	Logical parameter .T. cause video to pause, .F. to restart if paused
Play	Numeric parameter indicating at what second of the video to begin playing
Refresh	Repaints the control and updates any values
Stream	Allows a video to be loaded from a memory variable

Events

Event	Description
Destroy	Executed before a control is released from memory
Error	Executed if an unhandled error occurs
Init	Executed when a control is created
Loaded	Executed after a video file is loaded

Example

Notes

See Also

DRAFT

DRAFT

JAXBase Concepts

The following chapters are designed to help you understand how JAXBase can be used in various use cases and attempt to clarify any lingering questions about the various subjects.

DRAFT

Arrays, Variables, and Objects

DRAFT

Data Sessions and Work Areas

DRAFT

DBF Fields

DRAFT

Dates

DRAFT

Debugging and Error Handling

DRAFT

Using Indexes

DRAFT

Using the Keyboard Commands

DRAFT

Merging Text

DRAFT

Moving Data To and From a SQL Database

DRAFT

Code Page Support

DRAFT

Reading and Writing Barcodes

DRAFT

Tables and Cursors

DRAFT

Using One or More a SQL Database

DRAFT

Using the Regular Expression Object

DRAFT

DRAFT

Appendices

DRAFT

Appendix A – File Types

The recognized file types in JAXBase are as follows:

PRG	Source JAXBase code
JXP	Compiled JAXBase code
APP	One or more compiled JAXBase code files used to create an application
DEF	Class Definition source code
JXD	Compiled Class Definition code
H	Include file used during compiling of source code
DBF	Data table
FPT	Data table memo file
JST	Data table JSON file (64-bit tables only)
IDX	Simple data table index
VCX	Class Library
VCT	Class Library memo file
JXV	Compiled class library
JXVP	Intermediate JAX source file for class library
SCX	Form / Formset definition
SCT	Form / Formset memo file
JXS	Compiled Form/Formset
JXSP	Intermediate JAX source file for form/formset definition
FRX	Report definition
FRT	Report memo file
JXF	Compiled Report code
JXFP	Intermediate JAX source file for report definition
MNX	Menu Definition
MNT	Menu memo file
JXM	Compiled menu definition
JXMP	Intermediate JAX source file for menu definition
PJX	Project File
PJT	Project memo file
RPX	Report Definition
RPT	Report memo file
JXR	Compiled Report code
JXRP	Intermediate JAX source file for report definition
LBX	Label Definition
LBT	Label memo file
JXL	Compiled Library code
JXLP	Intermediate JAX source file for label definition

Appendix B – Table File Structure

32-bit Table Structure

Byte Offset	Description
0	File type: 0x01 FoxBASE: 0x02 FoxBASE+/Dbase III plus, no memo: 0x2F Visual FoxPro: 0x30 Visual FoxPro, autoincrement enabled: 0x31 Visual FoxPro, Varchar, Varbinary, or Blob-enabled: 0x42 JAXBase, VarChar, Varbinary, Blob-enabled: 0X?? dBASE IV SQL table files, no memo: 0x62 dBASE IV SQL system files, no memo: 0x82 FoxBASE+/dBASE III PLUS, with memo: 0x8A dBASE IV with memo: 0xCA dBASE IV SQL table files, with memo: 0xF4 FoxPro 2.x (or earlier) with memo: 0xFA
1 – 3	Last update (YYMMDD)
4 – 7	Number of records in file
8 – 9	Position of first data record
10 – 11	Length of one data record, including delete flag
12 – 27	Reserved
28	Table Flags 0x01 file has a structural .cdx 0x02 file has a Memo field 0x04 file is a database (.dbc) This byte can contain the sum of any of the above values. For example, the value 0x03 indicates the table has a structural .cdx and a Memo field.
29	Code Page Mark
30-31	Reserved, contains 0x00
32 – n	Field sub-records: The number of fields determines the number of field sub-records. One field sub-record exists for each field in the table.
n+1	Header record terminator (0x0D)
n+2 to n+264	If a VFP table, it may contain a backlink to a database container. JAXBase can read tables associated with VFP but will open the table as READONLY. If a JAXBase table, then these bytes are reserved.

32-bit Table Field Sub-Records

Byte Offset	Description
0 - 10	Field name padded with 0x00
11	Field type: W - Blob C - Character / Character binary Y - Currency B - Double D - Date T - DateTime F - Float G - General I - Integer L - Logical M - Memo / Memo binary N - Numeric P - Picture Q - Varbinary V - Varchar / Varchar binary O - System Field 1 - Null Tracking field
12 - 15	Displacement of field in record
16	Length of field in bytes
17	Number of decimal places
18	Field Flags: 0x01 System Column 0x02 Column can store null values 0x4 Binary column 0x0C Auto incrementing
19 - 22	Value of autoincrement Next value (Slated for Version 2)
23	Value of autoincrement Step value (Slated for Version 2)
24 - 31	Reserved

64-bit Table Structure

(Slated for Version 2)

Byte Offset	Description
0	JAXBase, VarChar, Varbinary, Blob-enabled: 0X??
1 – 3	Last update (YYMMDD)
4 – 11	Number of records in file
12 – 13	Position of first data record
14 – 15	Length of one data record, including delete flag
16 – 27	Reserved
28	<p>Table Flags</p> <p>0x01 file has a structural .cdx</p> <p>0x02 file has a Memo field</p> <p>0x04 file is a database (.dbc)</p> <p>This byte can contain the sum of any of the above values. For example, the value 0x03 indicates the table has a structural .cdx and a Memo field.</p>
29	Code Page Mark
30-31	Reserved, contains 0x00
32 – n	Field sub-records: The number of fields determines the number of field sub-records. One field sub-record exists for each field in the table.
n+1	Header record terminator (0x0F)
n+2 to n+264	<p>If a VFP table, it may contain a backlink to a database container. JAXBase can read tables associated with VFP but will open the table as READONLY.</p> <p>If a JAXBase table, then these bytes are reserved.</p>

64-bit Table Field Sub-Records

(Slated for Version 2)

Byte Offset	Description
0 - 40	Field name padded with 0x00
41	Field type: W - Blob C - Character / Character binary Y - Currency B - Double D - Date T - DateTime F - Float G - General I - Integer J - JSON Text Field L - Logical O - Long Integer M - Memo / Memo binary N - Numeric P - Picture S - Timestamp Q - Varbinary V - Varchar / Varchar binary 0 - System Field 1 - Null Tracking field
42 - 45	Displacement of field in record
46	Length of field in bytes
47	Number of decimal places
48	Field Flags: 0x01 System Column 0x02 Column can store null values 0x4 Binary column 0x0C Auto incrementing
49 - 56	Value of autoincrement Next value
57	Value of autoincrement Step value
58 - 63	Reserved

DRAFT

Appendix C – JAXBase Field Types

Code	Data Type	Description
B	Double	
C	Character	
D	Date Only	
F	Float	
G	General	
I	Integer	
J	JAX Text Field*	Proposed fixed length text field made up of 1 to 254 blocks of 128 bytes or 128 to 32,512 bytes in length. (Slated for Version 2)
L	Logical	
M	Memo	
N	Numeric	
Q	Binary Varchar	Max of 254 bytes in the 32-bit version Proposed 8,192 bytes max in 64-bit tables*
T	DateTime	
V	Varchar	Max of 254 bytes in the 32-bit version Proposed 8,192 bytes max in 64-bit tables*
W	Blob	
Y	Currency	

* Not available in Version 1

APPENDIX D – JAXBase Data Types

DRAFT

Appendix E – System Variables

DRAFT

Appendix F – JAXBase Data Sessions

DRAFT

Appendix G – DEF Files

A DEF file is like an H file in that it only contains definitions. In this case, class definitions, written in pure JAXBase code, that will be used by JAXBase to create user defined class, such as forms and components during runtime.

A DEF file can contain any number of class definitions which can be found by JAXBase by referencing those files using the SET DEFINITION TO command.



Appendix H – Collation Sequences

JAXBase Collation Codes higher than 1 are only valid with backend SQL Engines

Value	Description
0	xBase Standard UTF8, Case Sensitive, Accent Sensitive, Width-Insensitive
1	English (US) UTF8, Case Insensitive, Accent Sensitive, Width-Insensitive

Translation to Other SQL Engines

DRAFT

JAXBase FAQ

If you have questions about JAXBase, feel free to write to JAX@

Q: Why did you get rid of all those legacy commands? My application needs to be rewritten!

A: Such commands lend themselves to poor user experiences. I am sorry if you have been using a program developed in the last century, but much has changed since then. If you do not think JAXBase is right for you, then please feel free to continue using what you have. However, if being able to move to a powerful open source xBase system that works on both Linux and Windows, then is that not a small price to ask you to update your application?

Q: Why don't you support structural indexes and database containers?

A: I would say, "see the last question", but it is a bit more complicated than that. The database container and structural index in Visual FoxPro, for instance, are incredibly powerful and have features that most SQL backends do not directly support. Further, the local database container was a band-aid to the larger problem of blown indexes and corrupted tables. Finally, DBF tables have a limit of 2-gigabytes and JAXBase is being designed to consider 2GB of data as a "good start".

Since JAXBase supports both SQL Server and MySQL, and in the future will support PostgreSQL and other database back ends, it was decided to keep local DBF tables for applications that do not use large data sets.

Q: Why were reports and labels made into classes?

A: Great question! The report system was originally developed under MS-DOS and support for Windows was done by creating wrapper applications or other... well... "kludges" is the word I am hesitating to use.

Why should JAXBase build upon an MS-DOS-based reporting system and not take advantage of open-source solutions? Eliminating the in-language report preview rendering engine eliminates needless overhead and makes it easier to offer a cross-platform solution. Finally, using a cross-platform open-source office suite allows the creation of WYSIWYG reports that can be saved in different file formats.

Q: I notice there is a Barcode class. What is that about?

A: Barcodes are used extensively in industry and commerce and the lack of direct support in the xBase language is a stark reminder of its age and lack of advanced thinking by those who brought it into the new century. JAXBase will support reading and writing barcode images via the user's choice of various open-source or commercial solutions.

Q: What happened to the ON ERROR command?

A: Learn about the TRY/CATCH command to capture errors or use the Error method if dealing with a class. Ad-hoc error handling is so last century.