FUNCTION/OPERATOR SYNTAX

`X Y`	left and right arguments of a function/operands of an operator – any array
`M N`	– numeric array
`I J`	– integer array
`A B`	– Boolean array
`C D`	– character array
`f g h`	functions
`⍺ ⍵`	left and right arguments of a function train
`NS`	name or reference to namespace
`[ax]`	indicates functions that can have an axis specified
`[ct]`	indicates a dependency on ⎕CT/⎕DCT
`s/v/m`	indicates highest rank allowed is that of a scalar/vector/matrix

SELECTED ABBREVIATIONS

actions	⎕NQ action: 0 add to queue, 1 process immediately, 2 perform default action, 3 invoke OLE method, 4 signal ActiveX event
ax_mx	three-column matrix containing userID, aggregated file operation numbers and permission numbers
bytes	byte count
cn	component number
conargs	constructor arguments
dir|file	the name of a directory/file
etype	type of new object; one of ∇ (function/operator, the default value), ∊ (vector of character vectors), − (character matrix), ⍟ (namespace script), → (simple character vector), ○ (class script) and ∘ (interface)
name	the name of a variable, function or operator in the active workspace
nvpairs	one or more name/value pairs
object|ns	a *name* or a *ref*
pn	component file pass number
pnames	character scalar or vector containing file property names
ref	a reference to a namespace or object
regex	a Perl-Compatible Regular Expression (PCRE)
rw	read or read/write
tdno	thread number
tn	tie number for files; use 0 to generate number on tie/create
trans	transformation function or numeric codes to apply to matched expressions
type	internal data type – see TYPE CODES below

TYPE CODES

Constructed by prefixing one of the following numbers with the number of bits per element:
0 Unicode char, 1 Boolean, 2 Classic (⎕AV based) char, 3 Integer, 5 Floating point,
6 Pointer to Object or Nested Array, 7 Decimal floating point, 9 Complex.

Examples: 80 = 1-byte Unicode char, 163 = 16-bit integer, 645 = double-precision float
N.B. Pointers are reported as 326 in both 32-bit and 64-bit systems

NAME CLASSES (⎕NC and ⎕NL)

	2 Array	3 Functions	4 Operators	9 Spaces
.1	2.1 Variable	3.1 Traditional	4.1 Traditional	9.1 Namespace
.2	2.2 Field	3.2 dfns	4.2 dops	9.2 Instance
.3		3.3 Derived/Primitive	4.3 Derived/Primitive	
.4				9.4 Class (OO)
.5				9.5 Interface (OO)
.6	2.6 External/Shared	3.6 External		9.6 External class
.7				9.7 External interface

PRIMITIVE FUNCTIONS

SCALAR FUNCTIONS

Scalar functions are pervasive, apply item-wise and, when dyadic, respond to the axis operator

MONADIC

Syntax	Result	Implicit Args
`+Y`	Conjugate ('Identity' if Y not complex)	
`-N`	Negate: `0-N`	
`×N`	Direction ('Signum' if Y not complex)	
`÷N`	Reciprocal: `1÷N`	⎕DIV
`⌊N`	Round down to integer	[ct]
`⌈N`	Round up to integer	[ct]
`|N`	Magnitude (absolute value)	
`*N`	e raised to the power N	
`⍟N`	Natural logarithm of N	
`○N`	pi times N	
`!N`	Factorial (Gamma function of `N+1`)	
`?J`	Random number selected from `⍳J` (when J=0, a real number from <0,1>)	⎕IO, ⎕RL
`~B`	Logical Inverse: `0=B`	

DYADIC

Syntax	Result	Implicit Args
`M+N`	Add N to M	
`M-N`	Subtract N from M	
`M×N`	Multiply M and N	
`M÷N`	Divide M by N	⎕DIV
`M|N`	Residue after dividing N by M	[ct]
`M*N`	M raised to the power N	
`M⍟N`	Base-M logarithm of N	
`M⌈N`	Maximum of M and N	
`M⌊N`	Minimum of M and N	
`I○N`	Circular functions[1]	
`M!N`	Number of selections of size M from N (Beta fn)	
`M∧N`	Lowest Common Multiple of M and N	[ct]
`M∨N`	Greatest Common Divisor of M and N	[ct]
`< ≤ ≥ >`	Numeric comparisons[2]	[ct]
`= ≠`	General comparisons[2]	[ct]
`∧ ∨ ⍲ ⍱`	Boolean functions[3]	

[1] Circular functions (angles in radians)

`(-Is)○N`	`Is`	`Is○N`
`(1-N*2)*.5`	0	`(1-N*2)*.5`
`Arcsin N`	1	`Sin N`
`Arccos N`	2	`Cos N`
`Arctan N`	3	`Tan N`
`(N+1)×((N-1)÷N+1)*.5`	4	`(1+N*2)*.5`
`Arcsinh N`	5	`Sinh N`
`Arccosh N`	6	`Cosh N`
`Arctanh N`	7	`Tanh N`
`-8○N`	8	`(-1+N*2)*.5`
`N`	9	`<real N>`
`+N`	10	`|N`
`N×0J1`	11	`<imaginary N>`
`*N×0J1`	12	`<phase of N>`

[2] Comparisons

Comparisons return:
- 1 if proposition is true
- 0 if proposition is false

[3] Boolean functions

`A←1 0 0 1`
`B←1 0 1 0`

`A∧B`	1	0	0	0
`A∨B`	1	0	1	1
`A⍲B`	0	1	1	1
`A⍱B`	0	1	0	0
`~B`	0	1	0	1

PRIMITIVE FUNCTIONS continued

NON-SCALAR FUNCTIONS

NON-SCALAR MATHEMATICAL

Syntax	Result	Implicit Args
`⌹Nm`	Matrix inverse of **Nm** (square **Nm**)	
`⌹Nm`	Matrix pseudo-inverse of **Nm** (over-determined **Nm**)	
`Mm⌹Nm`	Multiply **Mm** with inverse of **Nm**	
`M⊤N`	Encode value N in number system M	
`M⊥N`	Decode: Evaluate N in number system M	

ARRAY PROPERTIES

Syntax	Result	Implicit Args
`⍴Y`	Shape: Length of each axis of Y	
`≡Y`	Depth: Maximum level of nesting in Y (-ve if uneven)	⎕ML
`≢Y`	Tally: Number of items in leading axis	

STRUCTURAL

Change structure, typically keeping all items

Syntax	Result	Implicit Args
`⊂Y`	Enclose: Scalar containing Y	[ax]
`⊆Y`	Nest: Y if already nested, else scalar containing Y	
`↑Y`	Mix: Remove nesting (⎕ML 1)	⎕ML, [ax]
`↓Y`	Split: Nest sub-arrays	[ax]
`∊Y`	Enlist: Simple vector from elements of Y (⎕ML 1)	⎕ML
`,Y`	Ravel: Reshape into a vector	[ax]
`⍪Y`	Table: Reshape into 2-dimensional array	
`⌽Y`	Reverse last axis of Y	[ax]
`⊖Y`	Reverse leading axis of Y	[ax]
`⍉Y`	Transpose: Reverse order of axes of Y	
`Iv⍴Y`	Reshape Y to have shape `Iv`	
`I⌽Y`	Rotate vectors along last axis of Y	[ax]
`I⊖Y`	Rotate vectors along leading axis of Y	[ax]
`Iv⍉Y`	Reorder the axes of Y	⎕IO
`X,Y`	Catenate: Join along last axis	[ax]
`X⍪Y`	Catenate First: Join along leading axis	[ax]

INDEX GENERATORS

Syntax	Result	Implicit Args
`⍳Jv`	Indices of all items of array of shape `Jv`	⎕IO
`⍸B`	Indices of all 1s in B	⎕IO
`⍋Y`	Upgrade: Indices to reorder Y ascending	⎕IO
`⍒Y`	Downgrade: Indices to reorder Y descending	⎕IO
`X⍳Y`	Index of: Indices in X of items of Y	⎕IO, [ct]
`X⍸Y`	Indices of items of Y in intervals with cut-offs X	⎕IO
`Is?Js`	Deal: `Is` distinct items from `⍳Js`	⎕IO, ⎕RL
`C⍋D`	Upgrade using collation sequence C	⎕IO
`C⍒D`	Downgrade using collation sequence C	⎕IO

SET FUNCTIONS

Syntax	Result	Implicit Args
`∪Yv`	Unique: Distinct items of `Yv`	[ct]
`X∊Y`	For each item of X, 1 if found in Y, else 0	[ct]
`X⍷Y`	Occurrences of entire array X within Y	[ct]
`X≡Y`	Match: 1 if X is identical to Y, else 0	[ct]
`X≢Y`	Not Match: `~X≡Y`	[ct]
`Xv~Y`	Without: `(~Xv∊Y)/Xv`	[ct]
`Xv∪Yv`	Union: `Xv,Yv~Xv`	[ct]
`Xv∩Yv`	Intersection: `(Xv∊Yv)/Xv`	[ct]

PRIMITIVE FUNCTIONS continued

SELECTION

Select items from an array

Syntax	Result	Implicit Args
`⊃Y`	First item of Y (⎕ML 1)	⎕ML, [ax]
`Iv⊃Y`	Reach into Y along path given by `Iv`	⎕IO
`Iv⌷Y`	Index Y using indices `Iv`	⎕IO, [ax]
`Iv↑Y`	Take `Iv` items along axes of Y	[ax]
`Iv↓Y`	Drop `Iv` items along axes of Y	[ax]
`Iv/Y`	Replicate along last axis of Y	[ax]
`Iv⌿Y`	Replicate along leading axis of Y	[ax]
`Iv\Y`	Expand last axis of Y	[ax]
`Iv⍀Y`	Expand leading axis of Y	[ax]
`Av⊂Y`	Partitioned enclose of Y according to `Av` (⎕ML 1)	⎕ML, [ax]
`Mv⊆Y`	Partition Y according to `Mv`	[ax]

DATA CONVERSION

Syntax	Result	Implicit Args
`⍎Dv`	Execute: Result of expression `Dv`	
`⍕Y`	Format: Character representation of Y	
`NS⍎Dv`	Execute `Dv` within namespace `NS`	
`Iv⍕Y`	Format Y using (width, decimals) pairs `Iv`	

IDENTITY FUNCTIONS

Return an argument unchanged

Syntax	Result	Implicit Args
`⌷Y`	Materialise items of Y in workspace	
`⊣Y`	Same: Y	
`⊢Y`	Same: Y	
`X⊣Y`	Left: X	
`X⊢Y`	Right: Y	

DFN SYNTAX

{⍺ function ⍵}		{⍺⍺ operator ⍵⍵}			
`⍺`	left argument	`⍺⍺`	left operand	`:`	guard
`⍵`	right argument	`⍵⍵`	right operand	`::`	error guard
`∇`	self reference	`∇∇`	self reference	`⍺←`	default left argument
				`1:s←`	shy result

FUNCTION TRAINS

```
 ( gh)⍵ →         g( hω)  ⍝ monadic atop
⍺( gh)⍵ →         g(⍺hω)  ⍝ dyadic  atop

 (fgh)⍵ → ( fω)  g( hω)  ⍝ monadic fgh fork
⍺(fgh)⍵ → (⍺fω)  g(⍺hω)  ⍝ dyadic  fgh fork

 (Xgh)⍵ →        Xg( hω)  ⍝ monadic Xgh fork
⍺(Xgh)⍵ →        Xg(⍺hω)  ⍝ dyadic  Xgh fork
```

PRIMITIVE OPERATORS

MONADIC

Syntax	Result
`{Is}f/Y`	Reduce: `f` between all items of Y (in groups of `Is`) on last axis
`{Is}f⌿Y`	Reduce First: `f` between all items of Y (in groups of `Is`) on first axis
`f\Y`	Scan: `f` between items of Y in progressively longer vectors along last axis
`f⍀Y`	Scan First: `f` between items of Y in progressively longer vectors along first axis
`{X}f¨Y`	Each: `f` on items of Y or between items of X and Y
`Xf⌸Y`	Key: `f` on items of Y grouped by unique X values
`f⌸Y`	Key: `f` on first axis indices of Y grouped by unique Y values
`{X}f⍨Y`	Commute: same as `YfX` (or `YfY` if no X specified)
`{X}f&Y`	Spawn: `f` on Y (or between X and Y) in a new thread
`{X}(Ns⌶)Y`	I-beam: Call experimental system-related service `Ns`

DYADIC

Syntax	Result
`{X}(f⍤r)Y`	Rank: `f` on or between trailing rank-`r` subarrays
`(f⌺Jm)Y`	Stencil: `f` on (possibly overlapping) rectangles of Y
`{X}(f⍣g)Y`	Power: iterates `f` (or `X∘f`) on Y until condition `YgfY` (or `YgXfY`) is true
`{X}(f⍣Js)Y`	Power: `f` (or `X∘f`) on Y `Js` times
`Xf.gY`	Inner Product: `f` / `g` between trailing vectors of X and leading vectors of Y
`X∘.gY`	Outer Product: `g` between each item of X and every item of Y
`f∘gY`	Compose (I): `f` on the result of `g` on Y, that is, `fgY`
`Xf∘gY`	Compose (IV): `X∘f` on the result of `g` on Y, that is, `XfgY`
`X∘gY`	Compose (II): `g` between X and Y, that is, `XgY`
`(f∘Y₂)Y₁`	Compose (III): `f` between `Y₁` and `Y₂`, that is, `Y₁fY₂`
`{X}(f⍠Zv)Y`	Variant: `f` qualified by `Zv` on Y (or between X and Y)
`(X@N)Y`	At: use values in X to replace positions N in Y
`{X}(f@N)Y`	At: apply `f` (or `X∘f`) to modify positions N in Y
`(X@g)Y`	At use values in X to replace positions identified by Boolean mask (`gY`) in Y
`{X}(f@g)Y`	At: apply `f` (or `X∘f`) to modify positions identified by Boolean mask (`gY`) in Y

CONTROL STRUCTURES

```
:For var :In|:InEach ax ⋄ block ⋄ :EndFor
:Hold tkn ⋄ block ⋄ :Else ⋄ block ⋄ :EndHold
:If bx ⋄ block ⋄ :ElseIf bx|:Else ⋄ block ⋄ :EndIf
:Repeat ⋄ block ⋄ :Until bx ⋄ :AndIf bx|:OrIf bx
:Repeat ⋄ block ⋄ :EndRepeat
:Select ax ⋄ :Case val|:CaseList val ⋄ block ⋄ :Else ⋄
                                               block ⋄ :EndSelect
:Trap ecode ⋄ block ⋄ :Case ecode|:CaseList ecode ⋄ block ⋄
                                         :Else ⋄ block ⋄ :EndTrap
:While bx ⋄ block ⋄ :AndIf bx|:OrIf bx ⋄ block ⋄ :EndWhile
:While bx ⋄ block ⋄ :AndIf bx|:OrIf bx ⋄ block ⋄ :Until bx
:With ns ⋄ block ⋄ :EndWith
```

`block`	one or more APL statements to be executed
`ax`	an expression returning an array
`bx`	an expression returning a single Boolean value (0 or 1)
`ecode`	an integer scalar or vector containing the list of event codes to be handled
`ns`	a namespace within which actions will be performed
`tkn`	the tokens that must be acquired before the thread can continue
`val`	an expression to compare with the array returned by `<ax>`
`var`	one or more loop variable name

`:Continue` – start next iteration of surrounding `:For`, `:Repeat` or `While`

`:Leave` – terminate `:For`, `:Repeat` or `While`

`:Return` – equivalent to `→0`

SYSTEM COMMANDS

The following system commands produce lists of specific types of names in the current namespace:
`)CLASSES`, `)EVENTS`, `)FNS`, `)INTERFACES`, `)METHODS`, `)OBJECTS`, `)OBS`, `)OPS`, `)PROPS` and `)VARS`. All these accept a starting letter for the list as an optional argument.

Command	Description
)CLEAR	Clear active workspace
)CMD *cmd*	Execute *cmd* (*cmd* is mandatory on Windows, optional on UNIX)
)CONTINUE	Save active workspace as CONTINUE and terminate session
)COPY *ws {nms}*	Copy (selected) contents of workspace *ws* to active workspace
)CS *{space}*	Change current namespace
)DROP *{ws}*	Erase file containing workspace *ws*
)ED *{etype} nms*	Open Editor for named objects of type *etype*
)ERASE *nms*	Delete named objects from the active workspace
)HOLDS	List tokens currently held (acquired by :Hold)
)LIB *{folder}*	List workspaces either on WSPATH or in *folder*
)LOAD *{ws}*	Replace active workspace with workspace *ws*
)NS *{name}*	Create new global namespace called *name*
)OFF	Terminate the session
)PCOPY *ws {nms}*	As)COPY but does not overwrite existing objects
)RESET *{n}*	Reset state indicator and empty event queue/clear top *n* suspensions
)SAVE *{ws}*	Save active workspace, optionally with new name *ws*
)SH *cmd*	Synonym for)CMD
)SI *{n}* {-tid=*tdno*}	Display (first *n* /last if *n*<0 lines of) state indicator (for thread *tdno*)
)SIC	Synonym for)RESET
)SINL*{n}* {-tid=*tdno*}	Display (first *n* /last if *n*<0 lines of) state indicator (for thread *tdno*) with local names
)TID *{tdno}*	Switch to thread *tdno* or report current thread number
)WSID *{ws}*	Set or report the name of the active workspace
)XLOAD *{ws}*	Load workspace *ws* without executing ⎕LX

SYSTEM VARIABLES

STATE SETTINGS AFFECTING BEHAVIOUR OF PRIMITIVE FUNCTIONS

Possible Values	Default	Description
⎕CT ←*(0 to 2*¯32)*	1E¯14	Maximum ratio between two numbers considered equal
⎕DCT←*(0 to 2*¯32)*	1E¯28	(⎕CT/⎕DCT when ⎕FR = 645/1287 respectively)
⎕DIV←*0|1*	*0*	Set to 1 to return 0 on division by zero
⎕FR ←*645|1287*	*645*	Specifies the result type of floating-point computations
⎕IO ←*0|1*	*1*	Specifies whether array indices are counted from 0 or 1
⎕ML ←*0|1|2|3*	*1*	Degree of compatibility with IBM APL2 (from 0=low to 3=high)
⎕PP ←*int (1-34)*	*10*	Number of significant digits in the display of numeric output
⎕RL ←*seed RNG*	*⍬ 1*	Seed and Random Number Generator used by Roll/Deal to generate random numbers

SELECTED ERROR CODES

0 Any 1-999	11 DOMAIN ERROR	1000 Any 1001-1006
1 WS FULL	12 HOLD ERROR	1001 STOP VECTOR
2 SYNTAX ERROR	16 NONCE ERROR	1002 WEAK INTERRUPT
3 INDEX ERROR	18 FILE TIE ERROR	1003 INTERRUPT
4 RANK ERROR	19 FILE ACCESS ERROR	1005 EOF INTERRUPT
5 LENGTH ERROR	20 FILE INDEX ERROR	1006 TIMEOUT
6 VALUE ERROR	21 FILE FULL	1007 RESIZE
7 FORMAT ERROR	22 FILE NAME ERROR	1008 DEADLOCK
10 LIMIT ERROR	24 FILE TIED	

SYSTEM NAMES

TOOLS AND ACCESS TO EXTERNAL UTILITIES

captured_output← ⎕CMD *cmd*	Execute Microsoft Windows *cmd*
r← ⎕CSV *data*	Convert CSV data *data* to APL array
r←data {header} ⎕CSV *format_spec*	Convert APL array to CSV data
r←type ⎕DR *x*	Interpret internal representation as array of type *type*
type← ⎕DR *x*	Return internal data type (*type*) of *x*
r←{format_spec} ⎕FMT *x*	Convert *x* into character matrix according to spec
r←{flag} ⎕JSON *data*	APL array from (*flag*=0) or to (*flag*=1) JSON text
name←{type}{shape} ⎕MAP *file {rw} {offset}*	Associate *name* with mapped *file* (from *offset*)
{name}←{name} ⎕NA *fn_desc*	Associate *name* with external DLL function
r←{tn} (reg_ex ⎕S *trans) text*	Search *text* for PCRE *reg_ex* returning *trans* (optional *tn* to spool output to native file)
r←{tn} (reg_ex ⎕R *trans) text*	Replace *text* selected by *reg_ex* using *trans*
captured_output← ⎕SH *cmd*	Execute UNIX shell *cmd*
r←{encoding} ⎕UCS *vec*	map chars to/from Unicode code points
⎕USING ←*ns_specs*	Set search path for .NET Namespace(s)
valid nums←{seps} ⎕VFI *text*	Validate numeric input: returns Boolean validity mask and numeric vector of converted input
r←{xml_options} ⎕XML *data*	XML string ←conversion→ APL array

SESSION INFORMATION/MANAGEMENT

r← ⎕AI	User number, compute, connect, keying time (ms)
user_name← ⎕AN	User (login) name
⎕CLEAR	Clear the active workspace
{names} ⎕CY *file*	Copy (optionally selected) *names* from saved ws
{num}← ⎕DL *num*	Delay and return seconds actually delayed
⎕LOAD *file*	Load saved workspace
⎕OFF	Terminate the APL session
⎕PATH ←*nss*	Set search path for functions and operators (blank-separated list of namespace names)
{r}←{flag} ⎕SAVE *file*	Save active ws in *file* with (with stack if *flag*=0) 0 returned on reload of saved ws, else 1
⎕SE	Session object
numvec← ⎕TS	Current time (y m d h m s ms)

MANIPULATING FUNCTIONS AND OPERATORS

r←{selector} ⎕AT *names*	Syntactical attributes of named functions or operators
r← ⎕CR *name*	Source of function or operator as a character matrix
{names}←{types} ⎕ED *names*	Open one or more objects (*names*) in the editor
{boolvec}← ⎕EX *names*	1 if each name is now free for use, else 0
{r}← ⎕FX *cr|nr|vr|or*	Name of fn or op defined from its matrix, nested, vector or object representation (or failing line no)
{level} ⎕LOCK *name|ref*	Hide source and optionally disallow suspension
r← ⎕NR *name*	Source of function or operator as a nested vector
state|data←{settings} ⎕PROFILE *action*	"Profile" CPU or elapsed time consumption of code
names← ⎕REFS *name*	List the names referenced by a function
{linenos}←{linenos} ⎕STOP *name*	Enable/report the current state of stops for a function
{linenos}←{linenos} ⎕TRACE *name*	Enable/report the current state of tracing for a function
r← ⎕VR name	Source of function or operator as a simple vector

NAMESPACES AND OBJECTS

⎕BASE.name	Invoke the base class definition of *name*
class_hierarchy← ⎕CLASS *ref*	Class hierarchy for a *class|instance*
ref←{class|interface} ⎕CLASS *instance*	Extract specific interface to an *instance*
{old_ns}←{names} ⎕CS *ns*	Switch to a *ns*, optionally exporting *names*
old_df← ⎕DF *char_array*	Set the display form of the current space
{ref}←{flags} ⎕FIX *source*	Define objects from *source* (vector of vectors or file name starting with file://)
refs← ⎕INSTANCES *class*	Current instances of *class*
instance← ⎕NEW *class {conargs}*	Create an instance of *class*
{me|ref}←{name} ⎕NS *names|ns*	Create (optionally named) namespace copying *names* or contents of *ns* into it
script← ⎕SRC *ref*	The source code of a *ref*
ref← ⎕THIS	Reference to the current namespace

SYSTEM NAMES continued

COMPONENT FILE FUNCTIONS

cn←x ⎕FAPPEND *tn {pn}*	Append *x* to end of file (optional *passnumber*)
r← ⎕FAVAIL	1 if file system is available, else 0
cns←{fchk_opts} ⎕FCHK *file*	Inspect and optionally repair *file*
{tn}←file ⎕FCOPY *tn {pn}*	Create copy of named *file* tied to *tn*
{tn}←file ⎕FCREATE *tn {64}*	Create new component file
{cn}← ⎕FDROP *tn n {pn}*	Drop *n* components from start (*n*>0) or end (*n*<0)
{tn}←file ⎕FERASE *tn {pn}*	Erase exclusively-tied file
r← ⎕FHIST *tn {pn}*	Return tied file *tn*'s history (user/time of last operations)
{tn}← ⎕FHOLD *tn {pn}*	Hold tied file *tn* (can be a matrix of *tn {pn}*)
names← ⎕FLIB *folder*	List component files in the specified *folder*
names← ⎕FNAMES	Names of tied files in same order as ⎕FNUMS
tns← ⎕FNUMS	Vector containing tie numbers of tied files
r←pnames ⎕FPROPS *tn {pn}*	Current values of the named properties of file *tn*
r←nvpairs ⎕FPROPS *tn {pn}*	Set properties of file *tn* using name-value pairs
ax_mx← ⎕FRDAC *tn {pn}*	Read access matrix
r← ⎕FRDCI *tn cn {pn}*	Size in bytes, user and timestamp of last update to *cn*
r← ⎕FREAD *tn cn {pn}*	The array stored in component(s) *cn* in file *tn*
{tn}←file ⎕FRENAME *tn {pn}*	Rename exclusively-tied file
{cn}←x ⎕FREPLACE *tn cn {pn}*	Store array *x* in component number *cn*
{tn}←{bytes} ⎕FRESIZE *tn {pn}*	Compact exclusively-tied file *tn* and set its max size
r← ⎕FSIZE *tn {pn}*	First *cn*, next-free *cn*, used and max size in bytes
{tn}←ax_mx ⎕FSTAC *tn {pn}*	Set access matrix for file *tn*
{tn}←file ⎕FSTIE *tn {pn}*	Share-tie component file
{tn}←file ⎕FTIE *tn {pn}*	Exclusively tie component file
{tn}← ⎕FUNTIE *tns*	Untie one or more component files

STACK AND WORKSPACE INFORMATION

numvec← ⎕LC	Lines at which each function on stack is suspended
⎕LX ←*expression*	Expression to be executed when workspace is loaded
numvec← ⎕NC *names*	Class of each *name* (fractional if *names* is encl. vec)
names←{init_chars} ⎕NL *nums*	Active names of specified class(es), optionally filtered
space_references← ⎕RSI	The spaces from which functions on stack were called
⎕SHADOW *names*	Make *names* local to most recently invoked defined fn
names← ⎕SI	Vector of names of functions on the stack
bytes← ⎕SIZE *names*	Space consumed by code/data attached to names
r← ⎕STACK	Definition of each function on the stack
r← ⎕STATE *name*	Details of the usage of *name* at each level of the stack
bytes← ⎕WA	Workspace available (unused)
⎕WSID ←*name*	The *name* of the active workspace
names← ⎕XSI	Full names of functions on the stack

THREADS

tdno← ⎕TCNUMS *tdno*	The child thread numbers of the given thread numbers
tdno← ⎕TID	The number of the current thread
{tdno}←{0|1} ⎕TKILL *tdno*	Kill threads *tdno* and (default 1 is true) descendants
⎕TNAME←*{tdname}*	Report/set the name (*tdname*) of the current thread
tdno← ⎕TNUMS	Report the numbers of all current threads
tdres← ⎕TSYNC *tdno*	Wait for threads *tdno* to terminate and return results

SYNCHRONISATION

{tkval}←{timeout} ⎕TGET *tktype*	Remove tokens of types *tktype* from the token pool
tktype← ⎕TPOOL	Type of each token in the token pool
{tdno}←{tkval} ⎕TPUT *tktype*	Add tokens to pool and return any *tdno* this unblocks
tktype← ⎕TREQ *tdno*	List token types that threads *tdno* are waiting for

SESSION OR DEVICE INPUT/OUTPUT

⎕ ←*x*	Output *x* to the session
x← ⎕	Evaluate user input and return result
⍞ ←*x*	Output *x* to session without trailing newline
charvec← ⍞	Return one line of user input

SYSTEM NAMES continued

NATIVE FILE FUNCTIONS

{r}←{flags} ⎕MKDIR *dir*	Create or ensure existence of directory *dir*
{offset}←x ⎕NAPPEND *tn type*	Append *x* using internal representation *type*
{tn}←file ⎕NCREATE *tn*	Create file (*tn* =0 to generate *tn*)
{r}←{flags} ⎕NDELETE *file*	Delete *file* (if *flags* =1, tolerate 'file not found')
{tn}←file ⎕NERASE *tn*	Erase tied file *tn*
r← ⎕NEXISTS *file*	Return 1 if *file* exists, else 0
r←{encoding} ⎕NGET *file {flags}*	Read Unicode text *file* as lines into array *r*
t←{properties}[1] ⎕NINFO *tn|file*	Return values of *properties*[1] for file *tn|file* (⍠1 will expand wildcards in *file*)
{rarg}←lock {secs} ⎕NLOCK *tn {offset} {bytes}*	Change *lock* status of file region (0=unlock, 1=read, 2=write)(optional timeout in seconds)
file← ⎕NNAMES	Names of tied files in same order as ⎕NNUMS
tns← ⎕NNUMS	Tie numbers of tied files
r←{flags} ⎕NPARTS *file*	Splits *file* into path, base name and extension (normalising file if *flags*=1)
{bytes}←vec ⎕NPUT *file {flags}*	Write (overwrite if *flags*=1) lines to Unicode *file*
vec← ⎕NREAD *tn type n {offset}*	Read *n* items of specified *type* from file *tn*
{tn}←file ⎕NRENAME *tn*	Rename tied file to have name *file*
{end_offset}←x ⎕NREPLACE *tn offset {typ}*	Write *x* to file at *offset* as *type*
{tn}←{bytes} ⎕NRESIZE *tn*	Resize file to have specified size
bytes← ⎕NSIZE *tn*	Current size of file
{tn}←file ⎕NTIE *tn {file_mode}*	Tie a file in the specified mode
{tn}← ⎕NUNTIE *tn*	Untie one or more files

BUILT-IN GUI & COM SUPPORT

{r}← ⎕DQ *object*	Process events generated by *object*(s)
{exported}←{flags} ⎕EXPORT *nm*	Specify fns to be exported by an OLEClient
{r}←{action} ⎕NQ *object event {params}*	Enqueue an event for processing
{name}←{name} ⎕WC *type {ordered_props} {nvpairs} …*	Create an instance of a built-in type and set property values
r←{object} ⎕WG *props …*	The values of the *properties* of an *object*
children←{class} ⎕WN *parent*	Child objects (of *class*) of *parent*
{old_values}←{object} ⎕WS *nvpairs…*	Set the values of one or more specified properties of *object*

ERROR HANDLING

⎕DMX	Namespace containing details of most recent error in current thread
⎕EXCEPTION	Details of most recent .NET exception
{msg} ⎕SIGNAL *errno|{nvpairs}…*	Signal an error; *nvpairs* set ⎕DMX props
⎕SIGNAL 0	Reset error-related system constants
⎕TRAP ←*trap_spec*	Define error handling

SYSTEM CONSTANTS

⎕A	The letters from A to Z
⎕D	The digits from 0 to 9
⎕NULL	A reference to a null item

[1] ⎕NINFO: Values for Numeric Array *properties*

X	Property	Default
0	Name of file/directory	
1	Type	0
2	Size (in bytes)	0
3	Last modification time	7⍴0
4	Owner user ID	''
5	Owner name	''
6	Whether file/directory is hidden (1) or not (0)	¯1
7	Target of symbolic link (when Type is 4)	''

OTHER SYSTEM NAMES

A number of system names that are no longer recommended for use in new applications have not been listed. Similarly, not all cases/variants of the listed system names are included

DYALOG

The tool of thought for software solutions

Reference Card

Dyalog version 16.0 (released June 2017)

Documentation: http://docs.dyalog.com/
Online help: http://help.dyalog.com/

Position the cursor after any symbol or name and press F1 to view the online help (except in TTY mode)

UK: +44 1256 830 030 US: +1 202 525 7994
sales@dyalog.com or support@dyalog.com
http://www.dyalog.com/