

HYPERTALK GUIDE

REFERRING TO OBJECTS

-- An object mentioned in an <expr> can be anywhere in the

same stack; only go may mention cds|bgs of another stack,

and stack properties may refer to a non-open stack

-- Legal chunks are: char, word, item, line, fld | button, cd, bg

-- Ordinals are: number; first-tenth, last; mid; any;

next; prev; this

-- If text is hilitied in a field, there is a legal container

called the selection which you can use or manipulate

-- An object mentioned in an <expr> can involve me (meaning,

the object where the handler lives); or the target (meaning,

the object that received the message) [but target refers

to a field's contents]

-- Constants are true, false, up, down, empty, quote,

return, space, comma, colon, tab, formfeed, linefeed

-- Contents types (for is a) are number, integer, point,

rect, date, logical

PROPERTIES

can (mostly) be set; "the" not required (outside msg)

NB: You can always use a property or a chunk of a property as

an entity of the same type (e.g. a number) in any expression

(you do not have to get it first)

• = cannot be set; ¶ = can only be set (no get!);

* = reverts to default when handler ends;

§ = there is a shortcut command (see below)

GLOBAL (no "of"):

UserPrefs:

blindTyping, powerkeys, textArrows <boolean>

userLevel <num>

userModify <boolean> overrides locking to let changes be made,

but they are discarded on closeCard

Parsing:

itemDelimiter <char>

Display:

¶*cursor <name> watch, busy, hand, arrow, ibeam, cross, plus

(but nothing is to stop you from adding your own)

*dragSpeed <num> bigger is faster, except that 0 is fastest

longWindowTitles <boolean>

scriptTextFont <name>, scriptTextSize <points>,

traceDelay <ticks>

Suppression:

\$*lockScreen <boolean> won't prevent msg appearing!

*lockMessages <boolean> will prevent msg appearing!

*lockRecent, \$*lockErrorDialogs <boolean>

Calculation:

*numberFormat <str> 0 = must fill; # = may fill

def "0.#####"

\$Printing:

printMargins <rect> 72=1 inch;

printTextAlign left|center|right; printTextFont <name>;

printTextSize, printTextHeight in points;

printTextStyle <styleList>

\$Painting:

brush, linesize, pattern <num>;

centered, filled, grid, multiple <boolean>;

multiSpace, polySides <num>;

textAlign, textFont, textSize, textHeight, textStyle

Lists:

•stacksInUse <ret list>

- address <zone:machine:program> of HC on the network

- [long] version of HC

returns 4 packed 2-digit numbers: major vers, minor vers,

software state (80 = final), release

OF WINDOW:

Variable Watcher:

hBarLoc, vBarLoc <num>

Message Watcher:

hideldle, hideUnused <boolean>;

¶nextLine, ¶text <expr> for debugging

Palette:

- buttonCount <num>; •commands <ret list>

hilitedButton <num> hilites that button; 0 to turn all off

External Windows (Picture):

globalLoc <pt>, globalRect <rect>; dithering <boolean>;

scale <-5..5>; zoom <in|out>

- pictureWidth, pictureHeight <num>

External Windows and Cd Window:

scroll <pt> the pt that is to be the new topLeft

All:

- properties <comma-list>

for some reason, not with cd window

- owner

also a way to discover type; besides HyperCard,

typical owners are Picture and Palette, or any picture-

making external

OF FLD:

autoTab, dontSearch, dontWrap, fixedLineHeight,

lockText, showLines, wideMargins, sharedText <boolean>

style <transparent | opaque | rectangle | shadow | scrolling>

scroll <num> pixels, not lines or anything helpful like that!

OF BTN:

hilite, autoHilite, sharedHilite, showName <boolean>;

style <transparent | opaque | rectangle | shadow | roundRect |

checkBox | radioButton>; icon <name|num>;

OF FLD OR BTN:

textAlign, textFont, textHeight, textSize, textStyle

OF WINDOW, FLD, OR BTN:

\$loc <pt> topLeft of a window, centre of fld/btn;

rect <rect>; left, top, right, bottom <num>;

topLeft, botRight <pt>;

NB: measured by cd window, but cd window by screen

width, height <num>

can be set for cd window, but •not other windows

\$visible boolean;
OF CD:

\$marked <boolean>

NB: a card also has properties rect etc.; this is really

a stack feature, the maximal cd window dimensions
OF CD OR BG:

\$showPict <boolean>
OF WINDOW, FLD, BTN, CD, OR BG:

•id <num>

but a card's id starts with "card id", and there is a

long id that gives much more info (also short id)

•number <num> front to back or start to finish sequence
OF STACK need not be open!:

cantAbort, cantModify, cantPeek <boolean>

•size, •freesize <num>; •version

gives a five-item list: creator version HC, compactor

version HC, oldest modifier since compaction, most recent

modifier, time of most recent modification (for format of

first four, see global Version, above)
OF CD, BG, OR STACK:

cantDelete <boolean>
OF FLD, BTN, CD, BG, OR STACK:

script <str>
OF MENUITEM:

markChar <char>; checkMark <boolean>; Senabled <boolean>;

menuMsg <msg>
OF NEARLY ANYTHING:

name <str> (*cannot be set for windows)

for stacks and their objects the reported name includes an

identifying prefix (e.g. "card field"); long name gives much

more info, short name just the name

FUNCTIONS

cannot be set; "the" required before or () after

["the F of X" legal only if there is exactly one param]

Confusing, eh? e.g., why "the stacksInUse" is a (non-settable)

Property, but "the windows" is a (non-settable) Function, or why

"the number of" can be a Property OR a Function, is a Mystery!

MOUSE AND FIELD-TEXT SELECTION:

mouse <up | down>; mouseClicked <boolean>

mouseH, mouseV, mouseLoc <pt> current

clickH, clickV, clickLoc <pt> at last click

clickChunk, selectedChunk, foundChunk <"char x to y

of card|bkgn field z">; if x > y, cursor is after y

clickLine, selectedLine, foundLine <"line x of ...>;

clickText, selectedText, foundText <word | grouped text>;

selectedField, foundField <"card|bkgn field z">;

selectedLoc <pt> the left bottom (!) of the selected text

KEYS:

commandKey, optionKey, shiftKey <up | down>

TEXT:

length(str); offset(findStr,inStr) <num>

charToNum(char); numToChar(num)

CHUNKS AND OBJECTS:

number(<chunk expr> | <obj expr>)

the <chunk expr> must name a container too (including

"menuitems of menu"); the <obj expr> usually must not mention

any larger entity, and refers to current card/stack --

except that you can say "cds of <bg expr>"

MATH:

random(num) <num between 1 and given num>

annuity(rate, numPeriods); compound(rate, numPeriods)

abs(num); average(list); max(list); min(list)

sin(), cos(), tan(), atan();

sqrt(); exp(); exp1() $\exp(x)-1$; exp2() 2^{**x}

ln(); ln1() $\ln(x+1)$; round(); trunc()

GLOBAL AND ENVIRONMENT:

[abbr] | [long] date; [long] time; secs; ticks

sound Done if no sound is playing; else name of sound

tool <x tool>; menus, windows, programs <ret list>;

screenRect <rect>;

diskSpace, heapSpace, stackSpace, systemVersion

SPECIAL:

value(<expr>) forces extra level of evaluation

param(num); paramCount; params <comma-list>

param(0) is the message (handler name);

params lists all, starting with 0. These are needed only

if you don't know in advance how many params will arrive

SYSTEM MESSAGES

[About the hierarchy: only btns, flds, and cds are sent messages; but if unused they then travel to

cd, bg, stack, stacks in use, home, and HC;

also, if handler contains a go, then to new cd, bg, etc.
(You insert/remove a stack from the hierarchy with start|stop using stack <name>;
StacksInUse stack handlers are consulted newest first.)]

Sending Order for Complex Events:
startup/resume absolutely first
all destructions (closes and deletes) precede

all creations (news and opens)
all closes precede all deletes; all news precede all opens
all same things (e.g. new) are sent together;

if destructive, concerning card, bg, stack;

if constructive, concerning stack, bg, card

startup, quit
suspendStack, resumeStack is stack frontmost in HC?

NB not sent if you switch apps in MultiFinder; if your

stack needs to know if it is running in the background

it can check the suspended
suspend, resume

for launch/return of app from HC: but not sent under

multifinder / system 7
openStack, openBackground, openCard, openField
closeStack, closeBackground, closeCard,

exitField, closeField

exitField if no change (since openField), else closeField
newStack, newBackground, newCard,
newField, newButton
deleteStack, deleteBackground, deleteCard,

deleteField, deleteButton
mouseEnter, mouseWithin, mouseLeave flds & btns
mouseDown, mouseStillDown, mouseUp

flds, btns, and cards! hence a card can "be" a button
enterInField, ReturnInField
moveWindow, sizeWindow, close card window

NB: further information about all of the above can be

derived by checking the target

openPalette, openPicture,

closePalette, closePicture (params: name, id)
mouseDownInPicture, mouseUpInPicture (params: name, xy)
keyDown, commandKeyDown (param: char)

[keyDown distinguishes case; command doesn't]

keyDown not sent if commandKey is down; but it is sent before:
returnKey, enterKey, tabKey,

arrowKey (param: direction),

controlKey (param: ascii code),

functionKey (param: which)

idle

choose [the second param is the toolNum]

doMenu (params: itemName, menuName)

errordialog (param: errorMessage)

sent if error when lockErrorDialogs is True

appleEvent (params: class, id, sender)

...and commands (put, get, go, etc.) count as messages! (but keywords

such as send, repeat, etc., do not)

Handlers are of the form on <msg> [paramList]

Intercepted messages can be let thru with pass <msg>

You can (re)direct messages with the command

send <msg> to <obj expr>

(Observe that Hypercard is an object; hence you can force
handlers to be bypassed)

COMMAND SYNTAX

CONTAINER/PROPERTY MANIPULATION:

put <expr> [into | after | before <cont>] default is msg

delete <chunk of cont>

"put/delete" smart about chunks, e.g. will add/cut nec delims

add|subtract|multiply|divide ... to|from|by ...

the thing affected must be a <[chunk of] cont>, but what affects it

may be any <expr>

get <expr> = put <expr> into it

set <prop> [of <obj>] to <expr>

set property shortcuts:

reset paint properties to def

reset printing ditto

lock screen; unlock screen [with <visual>]

enable | disable [menuItem <name|num>] of menu <name|num>

lock | unlock error dialogs

(see also Showing and Hiding, Marking, below)

SIMULATING USER ACTIONS:

select [before | after] <chunk expr>|text of <fld expr>

NB select by word/line will not include the

space/cr after (must use char for that)

To just place the insertion pt after char n of a fld,

use "select char n+1 to n of..."

select <btn|fld expr>

choose <name> tool; choose tool <num>

click at <pt> [with <modkey>[, <modkey>[, <modkey>]]]

drag from <pt 1> to <pt 2> [with <modkey1>[, etc.]]

<modkey> = shiftKey | optionKey | commandKey

type <str expr> [with commandKey]

doMenu <menuItemText>[, <menuItem>] [without dialog]

you can still do this even if you have deleted a standard

menu, though you cannot in that case use command-type

create stack <name> [with <bg expr>] [in new window]

save <stackname> as <stackname2> need not be current stack

SHOWING & HIDING:

picture [<name>[,<type>[,<style>[,<vis?>[<bitDepth>]]]]]

<type> = file default, resource, clipboard

<style> = plain, zoom def, document, roundrect true windows;

windoid, roundrect, shadow, dialog

these sit in back of palette layer, right over card

(and all but windoid are immobile); but they don't move if

user moves card window!

<bitDepth> (1-32) causes offscreen buffer copy; 0 forces none
palette <name>[, <loc>]

NB you cannot precache a palette (but you can use offscreen loc)

-- Note that picture and palette create, and what they create

are thereafter windows
close <window expr>

<window expr> = cd window | window <name> | <num> | <id>
hide titlebar | menubar | <window|btn|fld expr>
show titlebar | menubar | <window|btn|fld expr> [at <loc>]

here, <window expr> includes std titles (msg, tool window, etc.)

NB loc for cd window is measured from screen corner
show|hide cd|bg pict; show|hide pict of <cd|bg expr>

the second type affects unseen cards in same stack
show|hide groups

MARKING, FINDING & SORTING:

[un]mark all cds | cds where <boolean expr> |

cds by finding <str> [in <fld expr>]
find [whole|word|chars|string] <str> [in <fld expr>]

a) spaces regarded as delimiting separate strings:

normal: there are words beginning w/ every given str

word: the given words all appear as words

chars: the chars that make up each str all appear

b) spaces regarded as part of a single str:

string: the full string appears

whole: the full string appears at word-boundaries
sort [marked] cds [of <bg expr>] [<how>] by <expr>
sort lines | items of <cont> [<how>]

<how> = [ascending | descending]

[text | numeric | international | datetime]

default is ascending and it figures out its own value

the by <expr> is what about each cd we are to evaluate
NB: no need to test for existence of <fld> in any of these!

DIALOGS:

answer <prompt> [with <btn1> [or <btn2> [or <btn3>]]]

default <btn1> is "OK"; last btn is hilited
answer file <prompt> [of type <type> [or <type2> [or...]]]

standard <type>s are stack, text, application, picture,

paint; otherwise use file type code
answer program <prompt> of type <type>

to choose from programs running now on the net
ask [file] <prompt> [with <default reply>]

you can distinguish an empty reply from Cancel; the result

will be Cancel if the user presses Cancel
ask password [clear] <prompt> [with <default reply>]

MENUS:

create menu <name>
delete [menuItem <name|num> of] menu <name|num>
put <itemList> before|into|after [menuItem <name|num> of]

menu <name|num> [with menuMsg <msgList>]

in <itemList> use "-" for grey line

a msg "doMenu ItemName" is always sent; if no doMenu handler

traps it, or if it traps but passes it, then the custom msg is

sent, if any; if none, HC responds with its internal handler. So,

for a standard menu, you can add something to the start of the

standard response, or just totally change the response
reset menubar

NAVIGATING:

go <cd | bg | stack expr [w/ stack: "in a new window"]>

The cd|bg expr may include stack expr for a different stack.

-- NB: numerical marked cd exprs are valid cd exprs (first

marked cd, marked cd 7); but a bug (?) causes such exprs

to default to the first cd even if no cds are marked. To get

around this, test for "next marked cd", which works correctly
push [this | recent meaning previous viewed] cd
pop cd [into | before | after <cont> fetches cd id and stack name
show <cardinal num> | all | marked cds

COMMUNICATING:

open file <name>

must open before you can read or write
read from file <name> until <char>|eof
read from file <name> [at <pos>] for <num> of chars
write <expr> to file <name> [at <pos>|eof]
close file <name> done reading or writing

print <expr> string or equivalent
open printing [with dialog] to let user pick format
open report printing [with dialog | template <name>]
print all | marked | <cardinal num> cds
print cd | <cd expr> [from <pt> to <pt>]
close printing

open [<doc> with] <app> launches
print <doc> with <app>
close <app>

careful! don't type Close "Finder" into msg box now!!! STOP!
close <doc> with <app> only if apple-event savvy
send <scriptname> to program <name> [without reply]

request <exprToEvaluate> from program <name>
request ae data [keyword <expr>]

for use upon receipt of an appleEvent; the default (no

keyword) is the "direct object" (keyword "----")
reply [error] <expr> [keyword <expr>] to the sender

import paint from file <name>; export paint to file <name>
dial <num> [with <modem commands>]

SPECIAL EFFECTS:
visual <effect name> [<speed>] [to <color>] next time cd changes

<effect name> = plain; zoom open / close from/to clickloc;

zoom in / out; iris open / close; barn door open /close;

wipe / scroll right / left / up / down; dissolve; checkerboard;

venetian blinds; shrink to/from top/center/bottom

<speed> = [very] slow | fast

<color> = black, white, gray, inverse, card
flash [<num flashes>]
beep [<num beeps>]
wait <num> [ticks|seconds]; wait until|while <boolean expr>
play <snd resource name> [tempo <speed>] ["<notes>"]

included are Harpischord, Flute, Boing; or play stop to kill

200 is a moderate tempo

notes are abcdefg [#|b] with octave (4=middle), or 60 = middle C

durations are whqest (whole-thirtysecond) [.|3]

duration and octave remain in effect until altered

NB: play <name> 0 = preload
edit script of <obj expr>
convert <date expr> to <date format>

formats are: seconds, long date, short date, abbreviated date,

long time, short time, dateItems

NEW (2.1) FEATURES

CONSTANTS:

comma, colon

WINDOWS:

properties are: loc, visible, name, number, ID, owner

OPENPICTURE and OPENPALETTE now send ID as well as name

GLOBAL PROPERTIES, FUNCTIONS:

the lockErrorDialogs, set by (un)lock error dialogs

(in this mode, errorDialog is sent to card on error)

the itemDelimiter (usually comma)

the address (of HyperCard on the network)

the programs (gives ret-list of running programs)

the systemVersion

READ and WRITE:

read/write at <BYTENUM>, where eof is a valid BYTENUM

read until eof (and the 16K limit is gone)

DIALOGS:

answer program <PROMPT> lets choose running progs on net

and of course, APPLE EVENTS...!

Sources: HyperCard Help Stack; HyperTalk Reference Stack;
Danny Goodman, The Complete HyperCard 2.0 Handbook;
John Kevin Calhoun, HyperCard 2.1 Release Notes;
and lots of poking around and experimenting