TealInfo DevKit Manual

Table of Contents

Introduction

Hello. Welcome to a brief tutorial to Making TealInfo Folios. This document assumes a good working knowledge of using unix-style command-line based utilities. It is also helpful to have some experience scripting HTML or writing macros or batch files.

TealInfo Folios are information/database applications you can make yourself with nothing more than the included tools and a text editor. You describe the Folio's display and functionality in the text file, then run the tool MkTlInfo to convert the text file into a Folio that can be synced into a handheld and browsed from the TealInfo application. The results you end up with are flexible interactive documents that look and feel like custom applications. That having been said, let's get started.

Contents

The contents of this package may vary from platform to platform. In general, it should contain:

MKTLINFO – folio creation program
UNTLINFO – folio decompiler program
README.TXT – list of files in the package
Example folio source files

Be sure to check the README file for a list of files and descriptions. Many example files are included, and one may help you with a specific application, be it company data display, catalogs, calculator, or data forms application.

Making Folios

Like TealDoc files, TealInfo folios are created first as text files that then get converted to a Palm-recognizable format using a conversion program. For TealInfo folios, the MkTIInfo converson tool is included for this purpose.

Getting Started

The best way to begin making Folios is to start with one of the included sample folios and modify it to suit your needs. Load it into a simple text editor like NOTEPAD to perform the editing. If you load it into a full-fledged word processor, make sure that you save any changes as a *plain text file*.

If you make changes you might want to simply comment out sections you change rather than deleting them so you still have the old versions to refer to. Do this by inserting a pound symbol (#) at the front of the line to comment that line out.

Running MkTIInfo

On a PC, run MkTIInfo from a DOS box with the following format:

```
MKTLINFO <input text> <output .PDB> <folio title> <image database>
```

If any of the file names or titles have spaces, they should be enclosed in quotation marks ("). The image database (optional) is the name of a TealPaint image database in the current folder to insert into the folio to be used as an embedded source of imagery.

You can also run MkTIInfo without arguments and you'll be prompted to input the necessary file names.

When making a folio, you'll no doubt need to quickly preview changes, particularly when positioning on-screen objects. We recommend using POSE, the Palm OS Emulator, available at Palm's web site http://www.palm.com. This program simulates a Palm organizer on your computer desktop, allowing you to nearly instantly load new version of a folio for quick previewing without having to HotSync the folio onto an actual handheld to test it.

For more platform-specific installation and running information, refer to the file README2.TXT that accompanies this archive.

Decompiling Folio Files

Use the tool UNTLINFO to unmake Folios, creating an editable sample file from a finished folio. Use this to take apart existing folios to use as examples in making your own, or as part of the process of retrieving information from a folio used in a forms application. On a PC, UNTLINFO is a command-line program whose command format is:

```
UNTLINFO <input .PDB file> <output text file>
```

For example, you might type:

```
UNTLINFO myfolio.pdb myfolio.txt
```

If you enter no arguments, the program will prompt you for the arguments one at a time. Furthermore, if the folio requires a password, you will be asked to enter it to disassemble

the folio. The disassembled folios will not be 100% copies of the original source text, but should be adequate to reproduce or remake most if not all of the functionality of the original.

Testing Folios

The obvious way to test Folios is to load it into a PalmOS handheld, but this is also one of the slowest. Instead, it's much more convenient to use a Palm OS Emulator like POSE, which is available free from www.palm.com. You can simply remake the folio and load it up immediately from the POSE popup menu and start TealInfo to run it.

Style Hints/Guidelines

Author Information

When making a folio, please include an About popup object indicating author information, distribution guidelines, and version date or number.

Object sizing and placement

Objects should be sized appropriately for their contents.   For the best use of space, windows and lists using Palm font 0 or font 1 should have their height set to:

(number of displayed lines) * 12 + 1

Objects should also be placed with regular spacing, lined up where appropriate. The easiest way to accomplish this is to use **relative placement tags**, described below.

Insure that popup lists align properly with their trigger boxes. In general, the text should be right justified and the right-side edges of both the popup and trigger should line up. Also, the popup should be no wider or taller than necessary to hold the list members.

Interface

Make appropriate use of the hardware page-up and page-down buttons to make folios more easy to navigate, using the BUTTON_SCROLL and BUTTON_SELECT style flags where appropriate. When using the STROKE_FIND feature, note that this feature is available via an on-screen note or at least in an informational popup.

Text Resources

To keep the size of folios down, make ample use of TRES text resources whenever you have large duplicated TEXT blocks in a Folio. Note that TRES resources can only replace an entire TEXT block. You cannot create a single TEXT block by combining multiple TRES resource. You can, however, create an object with multiple TEXT blocks, each referencing different TRES resources.

Max Object Size

Each object has a maximum size of 32 kilobytes, due to a limitation in the Palm OS record size. This may be a problem if that object is linked and therefore has many alternate states (multiple TEXT blocks). You can get around this limit by using TRES text

resources for each of the object's states, thereby splitting up the object's text into separate records.

Quotes

If you need to imbed quotation marks inside the quoted text of a TEXT block, insert them by placing two quotation marks ("") next to each other wherever the quote should appear.

Text Alignment

You currently have limited text-alignment options for multi-column tables. Basically, you have one basic alignment style flag (none=left, **ALIGN_RIGHT**, **ALIGN_CENTER**) for the whole table, and override flags, which, if specified, can set different alignment for the first and/or last columns individually (**ALIGN_START_LEFT**, **ALIGN_END_CENTER**, etc).

Horizontal Scrolling

By default, text within objects is formatted to fit within the width of the object. To create a horizontally-scrolling object, just use the TABS tag to create one or more columns with tab stops so that the total width of the columns exceeds the pixel width of the object. If you are not otherwise using columns, just use a TABS tag set to the virtual width of the oversized scrolling area.

Hardware Buttons

Using the BUTTON_SCROLL or BUTTON_SELECT style tags to link the scrolling or selection of an object to the PAGE-UP and PAGE-DOWN buttons or other four hardware buttons. These flags only have an effect on a non-popup object when no popups are active. When used on a popup object, it need not be activated to change with the button press.

Stroke Find

The STROKE_FIND style flag allows you to set a LIST or OUTLINE object's current selection by entering a graffiti stroke for the first letter of the desired selection. Each subsequent character will advance the current selection to the next line starting with that letter, looping back to the start after the last entry in the list.

Linking

Remember, if you are trying to link objects and are getting unpredictable results, only the first object in the list of links can have a variable number of entries. Typically, you run into problems if you have one object dependent on another (for say, a category and then a sub-category), and then create a third object dependent on the previous two. If this is the case, you typically need to use the MAXVAL tag in the second object to specify a fixed selection count for the math. See the "Using Object Links" section below for more instructions and details.

The remainder of this document is exerpted from the TealInfo User's Manual

4

MkTIInfo Source File Format

MkTIInfo takes plain text files with the following basic format: Objects are defined by a line with the name of the OBJECT, followed by lines containing FIELDS describing the object. Each field is followed by one or more VALUES on the same line, though TEXT fields may have VALUES on subsequent lines as well. VALUES can be numbers, expressions, keywords, or quoted text, depending on the type of FIELD.

```
(OBJECT1) (ID)
(FIELD1) (VALUE1A) (VALUE1B...)
(FIELD2) (VALUE2A) (VALUE2B...)
...

(OBJECT2) (ID)
(FIELD1) (VALUE1A) (VALUE1B...)
(FIELD2) (VALUE2A) (VALUE2B...)
...
```

where (OBJECT) is a keyword defining an on-screen interface element such as a list, label, or button, and (FIELD) is the name of an adjustable parameter for that object. Each (FIELD) is followed by one or more values, depending on the particular (FIELD).

Comment lines are indicated with a pound (#) sign. An lines beginning with a '#' are ignored by the folio converter.

EXAMPLE: This example creates a simple form with one scrolling object

```
#
# This is a simple scrolling list
#

LIST
    X 10
    Y 20
    W 140
    H 130
    TEXT    "The Man Trap"
            "Charlie X"
            "Where No Man Has Gone Before"
            "The Enemy Within"
            "Mudd's Women"
            "What are Little Girls Made of?"
            "The Naked Time"
    STYLE HRULE
    FONT 1
```

Relative Object Placement

When placing objects in a folio, it's often convenient to define an object's coordinates relative to another object, making them easy to align or move as a group. To help do so, numerical values can be entered as simple math expressions, and a number of predefined symbols can be used in the expressions, substituting for numbers:

```
X Y W H
```

> Coordinates of the current object (must be defined on a previous line to be used)

```
BX BY BW BH
```

> Coordinates of the current object's activation button

```
PREVX PREVY PREVW PREVH
```

> Coordinates of the previous object

```
PREVBX PREVBY PREVBW PREVBH
```

Coordinates of the previous object's activation button

For instance, to define a column of checkmark objects, one might write:

```
CHECKMARK
    X 10
    Y 15
    W 100
    H 12
    TEXT "One"
CHECKMARK
    X PREVX
    Y PREVY+12
    W PREVW
    H PREVH
    TEXT "Two"
CHECKMARK
    X PREVX
    Y PREVY+12
    W PREVW
    H PREVH
    TEXT "Three"
```

SPECIAL NOTE: mathematical expressions currently support five operators: addition (+), subtraction (-), multiplication (*), division (/), and modulus (%). Expressions are evaluated strictly from left to right, currently ignoring any mathematical precedence rules. Thus, 2+3*5 is evaluated as 25, not 17.

Special Tags

PNTFILE

While images for IMAGE and POPIMAGE objects can remain as separate TealPaint databases, it's often more convenient to ship a folio as a single file with the images imbedded inside. An entire TealPaint image database can be included inside a folio with PNTFILE tag. For example,

```
PNTFILE PICTURES.PDB
```

will imbed the image database "Pictures.pdb" (should be in the current folder) into a created folio. Any references to the picture database by IMAGE or POPIMAGE objects should then use the name of the folio as it appears on the PalmPilot instead of the image database name. Only one picture database can be imbedded in a folio.

OUTFILE and OUTNAME

MkTIInfo supports the use of special tags in the source file to pre-specify output and database (as seen on PalmPilot) file names. This latter name is especially important if the database is required to have a specific name because it's referred to by links from other folios or from internal references to imbedded TealPaint images.

Examples:

```
OUTFILE MYFOLIO.PDB
OUTNAME "My little folio"
```

Inline Font/Underline Selection

TealInfo supports changing fonts and underlining inside a body of text for a WINDOW or POPWINDOW object. Other objects may be less predictable. Font changing is done via HTML-like tags imbedded with the text of the object. For best results, tagged regions should not bridge lines. For example:

```
TEXT

    "This is some <$FONT=1>bold</$FONT> text."
    "This is some <$UNDERLINED>underlined</$UNDERLINED> text."
```

The <*$FONT*> tag selects a font, while the <*/$FONT*> tag reverts it back to the original font. Unlike HTML, tags of the same type cannot be nested, so a <*/$FONT*> tag will always revert back to the original font, even if preceeded by two <*$FONT* > tags.

Note that the original font defined in the object definition governs the spacing of text in a TealInfo object. Thus, you usually don't want to change fonts to one taller than the original font, or subsequent text lines will clip that text.

Inline Hyperlinks

You can also insert hyperlinks in text to open a named folio or activate a named button to pop up a window or picture. For example:

```
TEXT
    "Just tap <$LINK="My Folio">here</$LINK> to go to the index folio."
    "Just tap <$LINK="Info">here</$LINK> to show more information."
```

The link text can be either be the name of a folio to open or the name of a button to activate. The actual text of the button is used, not the ID of the button's object. Note that you can place the button entirely off-screen by setting it's BX field value to a large value, say 162, leaving the hyperlink as the only way to activate it.

Scrolling objects

TealInfo supports scrollable, oversized, text and image objects. The oversized width of text objects are defined by creating columns wider than the objects, while their heights are defined by the number of lines of content. Image widths and heights are defined by their source imagery.

```
WINDOW
        X 10
        Y 10
        W 50
        H 50
        TABS 75
        TEXT
                "This is an example of a text window with" \
                "horizontal scrolling"
```

Text objects appear with on-screen scrolling bars. In addition, for both types of objects, scrolling can be done by mapping the hardware up/down scrolling buttons and/or combinations for the four application launch buttons to vertical and/or horizontal movement. This is done using one of the BUTTON_SCROLL style flags.

By mapping scrolling behavior to different buttons for different objects, some folios can be made which operate largely pen-free.

Using Object Links

Links allow one to link the TEXT content of an object to the selections of one or more other objects. Basically, you need to create multiple alternate TEXT blocks for the object to be controlled, and then specify which other objects do the controlling.

To link an object, first identify the objects you're linking-to by giving each of them a unique ID name (with no spaces) following the object tag. For instance, the following identifies a list object as object 'choice_list'.

```
LIST choice_list
    X 5
    Y 5
    ...
```

Then, you can refer to this object in another object using the LINKS tag

```
WINDOW info
        LINKS   choice_list
        TEXT    "text one"
        TEXT    "text two"
        TEXT    "text three"
```

An object can only be linked to objects ("Linkees") which appear before it (the "Linker") in a text file. The "linker" object must also have multiple TEXT fields defined for it. The one that is actually used at any given moment is determined by the current selections of all of its "Linkees", multiplied out using the formula (example for 3 objects, 1, 2, and 3):

index = ((value1 * max2) + value2) * max3 + value3

where max1,max2,and max3 represent the respective number of selections objects 1, 2, and 3 can take on. In this example, you must have (max1 * max2 * max3) number of TEXT fields to cover all the possible values of value1, 2, and 3.

EXAMPLE:

```
LIST id_difficulty
    X 90
    Y 15
    W 60
    H 100
    TEXT
        "Easy"
        "Hard"
LIST id_speed
    X 10
    Y 15
    W 60
    H 100
    TEXT
        "Off"
        "Slow"
        "Med"
        "Fast"
WINDOW
    X 10
    Y 120
    W 140
    H 20
    LINKS id_difficulty id_speed
        TEXT "This is Easy and Off"
        TEXT "This is Easy and Slow"
        TEXT "This is Easy and Med"
        TEXT "This is Easy and Fast"
        TEXT "This is Hard and Off"
        TEXT "This is Hard and Slow"
        TEXT "This is Hard and Med"
        TEXT "This is Hard and Fast"
```

ADVANCED TIP: The maximum value for each Linkee is normally determined by that object's current contents. If the contents of a Linkee can change, the math will not come up predictably if it is not the first item linked, because the MAX value will be variable. This commonly occurs with two Linkees when the first Linkee is a category, and the second is a sub-category whose contents depend on the first Linkee's value.

```
LINKS palm_models options
```

If this is the case, use the MAXVAL style flag to fix a maximum value for that second LINKEE. Place the MAXVAL tag in the linkee's object definition, not in the object containing the LINK field referencing the linkee. Set MAXVAL equal to the real maximum number of selections the LINKEE need have, and place extra entries in the LINKER object to pad out the unused combinations, so that you have *max1* groups of *MAXVAL2* TEXT fields.

```
        LINKS palm_models options

        TEXT "1. Pilot 1000"
        TEXT "2. Pilot 5000"
        TEXT "3."
        TEXT "1. PalmPilot Personal"
        TEXT "2. PalmPilot Professional"
        TEXT "3."
        TEXT "1. PalmIII"
        TEXT "2. PalmIIIx"
        TEXT "3. PalmIIIe"
        TEXT "1. PalmV"
        TEXT "2. PalmVx"
        TEXT "3."
        TEXT "1. PalmIIIc"
        TEXT "2."
        TEXT "3."
```

For more info on how to do multiple dependencies, download our TEALINFO DEV KIT for examples.

Using Text Resources

Text Resources allow a folio to minimize duplicated text. Use the TRES tag to define an object with a single TEXT field. Be sure to give the TRES object a unique ID. Then, other objects can use this resource's text as its own object by simply referring to this ID preceeded by an '@' (at signed) instead of adding its own quoted text. Note that adding a TRES object uses about 50 bytes of overhead, so space savings will only occur with text of sufficient length, but text resource still allow easy editing of commonly-used text.

EXAMPLE:

```
    TRES id_notfound
        TEXT "Sorry, no information is available about this device."

    WINDOW
        X 5
        Y 14
        W 80
        H 100
        LINKS id_device id_class
        TEXT "4-5 Miles"
        TEXT "3 Miles"
        TEXT @id_notfound
        TEXT @id_notfound
        TEXT "15 Miles (estimated)"
```

Creating Commercial Folios

When creating a commercial folio, you may wish to use some of TealInfo's security features to keep your folio from being illicitly copied, decompiled, or used. As a first step, you can use the PASSWORD tag to lock the folio to a keyword. You can set up the keyword to be required to open the folio, or only if someone tries to decompile it. With the former option, using a numerical keyword or one that is not easily remembered, can keep the folio from being casually copied by someone unless they keep the keyword with them.

As a second level of protection, you create a custom version of the folio for every customer, using a key they wouldn't want to give out, like their credit card number.

Finally, if you collect the customer's HotSync user name, you can create a folio keyed to only work with their PalmPilot using the PASSWORD tag with with the LOCKOUT style option but an empty ("") TEXT field. See the TealInfo Object Reference in the Appendix for more information.

Using TealInfo for Forms Applications

With a little work, TealInfo can be used for simple forms applications using POPEDIT, EDITWINDOW, POPLIST, CHECKLIST, and CHECKMARK objects. For the latter three objects, the REGISTER style flag is

typically used to keep any choices permanent. To make EDITWINDOW changes permanent, the EDITWINDOW text must be stored in a separate TRES block,

To load the data back to the desktop, the folio "backup" option should be set in the details dialog for the folio. The folio will then be copied back to the user's Palm backup folder on HotSync. At that point, the folio decompiler "UntlInfo" (in the TealInfo Dev Kit) can be used to convert the folio back to text format suitable for parsing by a database script or converter program.

Future versions of TealInfo may include tools to help make such forms functionality available to general users and other non-programmers.

Using MATH Objects

One of the most powerful features of TealInfo is ability to create and use MATH objects. MATH objects display the results of a mathematical expression, which can combine data from other objects and their selections. MATH objects can be displayed or themselves be used as values for other MATH objects or LINK fields.

Expressions

Mathematical expressions can be specified in simple, intuitive form. They are currently evaluated strictly from left to right. Thus, the expression:

1+2*3

will evaluate to 9, instead of 7, which follows the accepted convention of performing multiply operations before additions and subtractions. To avoid confusion, parentheses should be used to specify left-to-right order when mathematical precedence would normally dictate otherwise.

(1+2)*3

The parentheses will be ignored, but this will maintain compatibility with future versions of TealInfo which may add support for both parentheses and precedence.

RPN Notation

Complex equations can also be specified using RPN (Reverse Polish Notation) popularized by HP calculators. The benefit of RPN is that it can evaluate any equation from left to right order without parentheses. While this document cannot go into a full explanation or tutorial of using RPN, the concept is simple:

RPN operates on the concept of a list of numbers, commonly called a "stack". When a number appears in an RPN expression it is simply added to the stack. When an operator (such as + or -) appears, it simply pulls the last two items off the stack, performs math on them, and places the result back onto the stack.

In evaluating a "normal" math expression (like 1+1), we normally start with a number, store it in our minds, then add an operator, remember it too, and then wait until we get another number to evaluate the expression. With complex expressions, we end up saving a whole bunch of numbers and operators and the order to evaluate them along the way to getting an answer.

RPN is much more efficient because you only store numbers. Operations occur immediately when you get the operator, so it's much more efficient. The operators appear in the order you would normally evaluate them, not the order they appear in the expression.

Some "standard" expressions translate to RPN as follows:

1+2	becomes	1 2 +
(1+2) -3	becomes	1 2 + 3 -
1+(2-3)	becomes	1 2 3 - +
(1+2)-(3+4)	becomes	1 2 + 3 4 + -

Operands

The numbers that appear in an expression, the "operands", can simply be constant numerical values such as 1, 2, 3.1415, or .000007.

Operands can also be the name of another object, preceded with an '@' sign or an '#' sign. If the object is a LIST or other selectable object, the '#' sign will return the current "value" (selected line number) of the object, substituted in for its name. The '@' sign will instead return the value of the object's text. If the object is a WINDOW, EDITWINDOW, other non-selectable object, or even another MATH object, then the object's current TEXT is evaluated as an expression itself and the result inserted into the current MATH object's expression. For LIST objects, the value of the current selection is used. For instance, a simple tip calculator could include:

TEXT "(@tip_rate * 0.01 + 1.0) * @sub_total"

Operators

MATH objects support a number of mathematical operators, including:

Floating Point Math

Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)

Integer Math (fractional parts of operators are ignored)
Modulus (%) – A%B is the leftover part after repeatedly subtracting B from A

Comparison (return 1 if true, 0 if false)
Greater-Than (>)
Less-Than (<)
Greater-or-Equal (>=)
Less-or-Equal (<=)
Equal (==)
Not-Equal (!=)

Logical (returns 1 if true, 0 if false)
Logical And (&&)
Logical Or (||)
Logical Xor (^^)

Bit Math
Bitwise And (&)
Bitwise Or (|)

Bitwise Xor (^)

Special
Truncate Digits (\) – truncates a number to a specified decimal place
Round Digits (:) – similar to truncate, but rounds up or down to the nearest result
Duplicate on Stack ($) – for RPN math only, duplicates the last entry on the stack

RPN Example

With a little ingenuity, MATH operators can be used to create some very neat folios. For instance, suppose you want to have a tip calculator which forces a 20% tip if the party is 6 or more people. The following RPN expression could take advantage of comparison operators to calculate a grand total in a tip calculator.

TEXT "@tiprate @numpeople 6 < * 20 @numpeople 6 >= * + .01 * 1.0 + @subtotal *"

@tiprate @numpeople 6 < *

The first operand of this expression just places the tipping rate on the stack. Then it compares the number of people into the party. Remember that comparison operators evaluate to 1 if true, 0 if false. Since this result is then multiplied with the tipping rate on the stack, it has the effect of zeroing out the total if 6 or more people are present.

20 @numpeople 6 >= * +

The next section, starting with "20", does the exact opposite. It evaluates to 0 if less than 6 people are present, or 20 otherwise. The two resulting products are added together, so the result is a rate of "tiprate" if fewer than 6 people are present or 20 otherwise. This is then multiplied by .01 to get a percentage, and added to 1 and multiplied with the subtotal to get the grand total.

Indexing

When using the '@' symbol to reference text values in LIST and TABLE objects, the current selection is normally used to index to the corresponding line or cell of the LIST or TABLE. Using brackets ([ ]), a specific entry can be manually specified instead. This allows LIST and TABLE objects to be used as convenient storage objects for lists of numbers used in calculations. The brackets should immediately follow the object name, with the index value enclosed by the brackets. For example,

```
@choice[1]
@choice[2]
@choice[3]
```

reference the first three items in a LIST named "choice". Interestingly,

```
@choice[#choice]
```

is equivalent to

```
@choice
```

Parameters

Sometimes, it is useful to re-use a mathematical expression defined in a referenced MATH object, but with one or two of the numbers changed from use to use. In these instances, curly braces ({ }) can be used to pass extra values into the MATH object to be used in evaluating the expression.

In the referencing MATH object, curly braces are added after the referenced object name, containing one or more values separated by columns. The values are then read from within the referenced MATH object using the special symbols $1, $2… etc.

For instance if a math object called "calc_tax" contains the following formula (rate/100+1)*subtotal:

```
TEXT "$2 100 / 1 + $1 *"
```

It can be used to calculate the totals for different subtotal/tax combinations:

```
TEXT "@calc_tax{21.95,6.5}"
```

Notes: Parameters can only be used in undisplayed referenced objects. As these objects expect and use parameters, they must be hidden using a blank PAGES tag, as their value without parameters is undefined. Also, when used in combination with **indexing**, parameters and their associated curly braces must come after the indexing square brackets, not before.

Special Notes

Math objects can also be used as sources for LINKs to other objects. When referenced as a LINK, only the integer portion (the part left of the decimal point) of the number is used. When referenced from an expression in another Math object, however, the full value is used.

Using DRAW Objects

DRAW and POPDRAW objects let you draw graphics onto a rectangular area of the screen with a series of simple graphic commands. Graphics generated this way can be much more space efficient that bitmap graphics, and can also vary according to user selections when combined with MATH objects.

DRAW object commands appear in a TEXT block with one command per line. Lines beginning with a '#' sign are ignored.

The following simple example draws a holiday greeting on the screen.

```
DRAW
        X       10
        Y       20
        W       140
        H       110

        TEXT
        "FCOLOR 212"
        "DLINE  40 90  70 20"
        "DLINE  70 20 100 90"
        "GLINE 100 90  40 90"
        "FCOLOR 161"
        "DRECT  60 60  7 7 3"
        "DRECT  70 30  4 4 3"
        "GRECT  50 80  7 7 3"
        "GRECT  80 70  6 6 3"
        "FCOLOR 255"
        "DRECT  65 90  10 4 0"
        "DFRAME 10 10 120 90 5 3"
        "DCHARS 1 30 40 Season's Greetings"
```

See the appendix for a full list of valid commands.

To substitute values from other objects into the command list use the '@' or '#' signs followed by the name of the object to insert. The '@' sign inserts the current text from the object or the numerical equivalent of that text if a number is needed. The '#' sign returns the current "value" of the object instead, which is the zero-based number of the current selection in an outline, list, or table.   The '#' sign will reference the integer portion of a MATH object as well.

<div align="center">Example: DLINE 10 10 @end_x @end_y</div>

DRAW objects expect numerical values only, and will **not** consistently evaluate expressions or external references with parameters on their own. When math operations are needed, DRAW objects should contain simple references to hidden MATH objects to perform the needed calculations.

Appendix A - MkTIInfo Object Reference

This appendix lists the screen objects and properties recognized by MkTIInfo in making custom TealInfo folios. More comprehensive references and examples can be found in the TealInfo Developer's Kit, found on the developer's section of the TealPoint web site.

Basic Display Objects

WINDOW

Description:

A basic screen object, WINDOW objects define a rectangular region of the screen which contains text. The text automatically wraps lines of text if necessary, and supports vertical or horizontal scrolling and optional columns and grid lines. Text in WINDOW objects can be viewed, but not be edited or selected.

Required Fields:

X Y W H TEXT

Optional Fields:

PAGES STYLE FONT TABS LINKS CYCLE DELAY COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL MASK_BUTTON_SCROLL

Notes:

A WINDOW holds a block of text which automatically word-wraps if longer than one line. Normally, to take advantage of this feature, paragraphs should be entered as one long line. As MkTIInfo only supports a 4000 character long line, this is typically done by using the continuation mark (\) with the TEXT fields. See the TealInfo field reference below for more information.

Columns and line rulings are supported, allowing the display of the text in tabular or columnar layout. To do so, use the TABS field to define the width of individual columns, and enter multiple quote-delimited chunks of data on each line, or separate the text for each column on a line using tab characters imbedded in the text. The VERT_GRID and HORIZ_GRID fields can be used to add grid lines to the columns. When VERT_GRID is defined, columns are treated as spreadsheet-like cells, and data is truncated if necessary to fit within each column. When omitted, columns are treated like tab-stops in a word processor; text may overflow into neighboring columns; tab characters simply advance to the next column.

By default, column entries are left-justified. The ALIGN_CENTER and ALIGN_RIGHT styles can override this justification in all columns. In turn, the ALIGN_RIGHT_START and ALIGN_RIGHT_END, ALIGN_CENTER_START and ALIGN_CENTER_END styles can be used to set alternate justification for only the first or last columns.

Since text in a window is not selectable, WINDOW objects have no "current value" and thus cannot be used as the target of a LINK field in another object. They can themselves LINK to other objects, however, and this is a common use of WINDOW objects. For instance, by defining multiple TEXT fields and using the LINK field to target, say, a LIST object, the window can be used to display different data depending on the current selection of the LIST object.

POPWINDOW

Description:

A scrollable, auto-wrapping text window that comes up when a trigger button is pressed.

Required Fields:

X Y W H BX BY BW BH BTEXT TEXT

Optional Fields:

PAGES STYLE FONT TABS LINKS BFONT COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER
ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END
ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL MASK_BUTTON_SCROLL FILLED
SQUARE_BUTTON

Notes:

A variation on WINDOW objects, most of a POPWINDOW object is normally hidden. Only a small trigger button defined by BX, BY, BW, and BH is shown on screen. The button resembles a standard PalmOS text button. A BTEXT field defines its label. When tapped, the window appears, and stays on screen until tapped to dismiss it.

LABEL

Description:

A simple text label.

Required Fields:

X Y TEXT

Optional Fields:

PAGES STYLE FONT COMPRESS

Supported Styles:

INVERTED ALIGN_CENTER ALIGN_RIGHT

Notes:

LABEL objects are drawn so that the upper-left, upper-right, or upper-center of the text is aligned at the designated coordinate. Text is not wrapped and cannot scroll.

RECT

Description:

A bare graphic rectangle, either filled or not, used as a graphic embellishment

Required Fields:

X Y W H

Optional Fields:

PAGES STYLE

Supported Styles:

FILLED ROUND_BORDER BOLD_BORDER

IMAGE

Description:

Displays a graphic in TealPaint image format or subrectangle of an image in the specified rectangle

Required Fields:

X Y W H SX SY DATABASE RECORD

Optional Fields:

PAGES STYLE LINKS CYCLE DELAY

Supported Styles:

NO_BORDER BOLD_BORDER BUTTON_SCROLL

Notes:

The bounding box for the image must be a multiple of 8 pixels wide. SX and SY define the offsets into the source image from which to grab the subrectangle. SX must also be a multiple of 8 pixels wide, and the source image must not be wider than 160 pixels.

DATABASE defines the name of the image database. It is case sensitive and must match the name of the Image Database as it appears in TealInfo or TealPaint. The image can also be tacked onto the end of the TealInfo folio using MkTIInfo. If the image is appended in this way, then DATABASE should hold the name of the folio instead. Note that the image will fail to draw if the folio is renamed.

RECORD defines the image number of the database to use. The first image in a database is image 0.

To create simple page flipping animation, set the CYCLE parameter equal to the number of frames of animation. TealInfo will step forward through the image databases, looping from image RECORD to image RECORD+CYCLE-1, stopping between from frames by an amount specified by DELAY, in tenths of a second.

POPIMAGE

Description:

A popup-window displaying a TealPaint image or subrectangle of an image.

Required Fields:

BX BY BW BY BTEXT X Y W H SX SY DATABASE RECORD

Optional Fields:

PAGES STYLE LINKS CYCLE DELAY BFONT

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT
ALIGN_CENTER FILLED ALIGN_LEFT_START ALIGN_RIGHT_START
ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END
NO_SCROLL MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTERL

Notes:

This object has the same image and usage restrictions as an IMAGE object. As a small bonus feature, however, color or grayscale images shown in a full-screen POPIMAGE objects will appear in 16 grey levels on a Palm V-style display (Palm V, Visor), even under OS 3.1, which does not otherwise support the 16-shade mode.

Interactive Selection Controls

LIST

Description:

A scrollable vertical list of selectable text items, supporting optional columns and vertical and horizontal ruled lines.

Required Fields:

X Y W H TEXT

Optional Fields:

PAGES STYLE FONT TABS LINKS DEFAULT MAXVAL COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER

Notes:

LIST objects are interactive elements. While they are defined and used like WINDOW objects, text does not automatically word-wrap, and any line in the LIST can be highlighted by the user by tapping on it.

This gives the LIST a current numerical "value" ranging from zero (the first item) to the number of items in the list (minus one). This value can be used as an operand for expressions in MATH objects, or change the contents of other objects that have LINK fields referring to us.

The entries in a LIST are specified using multiple-lines of text in a TEXT field. Each line specifies a separate entry in the list. If the LINK field is used to link the object to another object's value, multiple TEXT fields can be used to define changing contents for the LIST.

POPLIST

Description:

A scrollable, selectable item list triggered by popup button, resembling the PalmOS category selector

Required Fields:

X Y W H TEXT BX BY BW BH

Optional Fields:

PAGES STYLE FONT TABS LINKS DEFAULT MAXVAL BFONT COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND FILLED REGISTER SQUARE_BUTTON

Notes:

The trigger button for POPLIST objects resemble standard PalmOS category selectors. They show the text of the current selection with a triangular trigger symbol. They should be sized wide enough to fit the widest text line to be drawn without truncating the text.

TABLE (New in 4.0)

Description:

Nearly identical to LIST objects, TABLE objects bring up a scrollable, selectable list of text items. Columns defined by the TABS field, however, are required. Unlike a LIST, which selects an entire row of items at a time, TABLE objects allow the selection of single cells within the object. A TABLE object with a single column is equivalent to a LIST object.

Required Fields:

X Y W H TEXT TABS

Optional Fields:

PAGES STYLE FONT LINKS DEFAULT MAXVAL COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER

Notes:

The current "value" of a TABLE object corresponds to the current highlighted cell. Cells are numbered starting at zero from left to right and then top to bottom.

POPTABLE (New in 4.0)

Description:

A pop-up version of the TABLE object, POPTABLE objects bring up a Table when their trigger buttons are tapped. The trigger buttons show the contents of the current cell.

Required Fields:

X Y W H TEXT BX BY BW BH TABS

Optional Fields:

PAGES STYLE FONT LINKS DEFAULT MAXVAL BFONT COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND FILLED REGISTER SQUARE_BUTTON

Notes:

The trigger button shows the name of the currently selected item, and BX BY BW and BH define its bounds. X, Y, W, and H define the bounds of the table the trigger brings up. The button bounds must be large enough to contain the longest cell in order for it to draw correctly.

OUTLINE

Description:

A scrollable, selectable, hierarchical item list, this object shows a list of data in outline tree form, where subsections of the tree may be collapsed and hidden from view when desired.

Required Fields:

X Y W H TEXT

Optional Fields:

PAGES STYLE FONT TABS LINKS DEFAULT MAXVAL COMPRESS

Supported Styles:

HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER NO_SCROLL
MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER

Notes:

Similar to standard LIST objects, OUTLINE objects show a list of selectable text items. Unlike a LIST, however, the items are displayed like an outline tree with items, sub-items, and sub-sub-items. Subsections can be hidden or shown by tapping on control triangles next to items in the outline.

To define an outline list's contents, create a multi-line TEXT field, preceeding lines with one or more '>' or '<' characters to define it's level in the tree. Root items are bare. Subitems to root entries are preceeded by one '>', sub-items to sub-items are preceeded by '>>', etc. If the item in the list should appear initially when the folio is first opened, then use a '<' character instead of a '>'. The '<' character forces the control on the previous line to default to the open state.

To set up an OUTLINE object for horizontal scrolling, use a TABS field with a single value specifying the virtual width of the scrolling area in pixels. OUTLINE objects not not otherwise support columns.

The following example sets up an OUTLINE object with three levels of items:

```
OUTLINE

    X 10
    Y 20
    W 100
    H 120
    TEXT "Restaurants"

        "Hotels"
        "Taxis"
        "Museums"
        ">Art"
        ">Nature"
        ">>Marine"
        ">>Land"
        "Nightclubs"
        ">Nice"
        ">Seedy"
```

POPOUTLINE (New in 4.0)

Description:

A POPOUTLINE object is pop-up version of an expanding OUTLINE object. When popped up, outline branches can be opened up or collapsed by tapping on standard outline control triangles. When tapping elsewhere on a line the list is selected and the popup window is closed.

Required Fields:

X Y W H BX BY BW BH TEXT

Optional Fields:

PAGES STYLE FONT TABS LINKS DEFAULT MAXVAL BFONT BTEXT COMPRESS

Supported Styles:

HORIZ_RULE INVERTED BOLD_BORDER NO_SCROLL MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER FILLED SQUARE_BUTTON

Notes:

See Outline objects for instructions on defining levels in a POPOUTLINE object.

CHECKMARK

Description:

A checkmark control resembling standard PalmOS checkbox controls.

Required Fields:

X Y W H TEXT

Optional Fields:

PAGES STYLE FONT LINKS TARGET DEFAULT COMPRESS

Supported Styles:

INVERTED REGISTER

Notes:

A simple one-item checkbox, CHECKMARK controls have a current "value" corresponding to their state. When used as a LINK target or MATH operand from another object, checkmark objects assume a value of 0 when unchecked and 1 when checked.

CHECKLIST

Description:

A scrollable, selectable list of text items, each line of which can have an optional checkbox.

Required Fields:

X Y W H TEXT

Optional Fields:

PAGES STYLE FONT TABS LINKS DEFAULT MAXVAL COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT
ALIGN_CENTER FILLED ALIGN_LEFT_START ALIGN_RIGHT_START
ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END
NO_SCROLL MASK_BUTTON_SCROLL BUTTON_SELECT STROKE_FIND REGISTER

Notes:

CHECKLIST objects are similar to LIST objects, but can optionally show checkboxes next to specified lines. The checkboxes can be checked on and off, and will be saved in the folio if the REGISTER field is defined. The current "value" of the CHECKLIST object, like standard LIST objects, is defined as the selected line, not by the state of the checkboxes.

List entries are defined by a multi-line TEXT fields, where each line of the text field defines an item in the scrolling list. Each line should start with either a plus (+) character for items that start up pre-checked or minus (-) character for those which start out unchecked. Lines with neither show up simply as labels.

Normally, the checklist will revert back to its initial state when the object is changed or the folio is exited. If the REGISTER style flag is used, the folio is not in flash memory, *and* the text for the checklist is stored in a TRES text resource, checked entries are permanently saved even if you leave TealInfo and return.

Data Entry Controls

EDITWINDOW

Description:

An editable text window. Often used to make folios for entry form or data acquisition.

Required Fields:

X Y W H TEXT

Optional Fields:

PAGES STYLE FONT TABS LINKS COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT
ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START
ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL
BUTTON_SCROLL

Notes:

An editable WINDOW object, EDITWINDOW objects are text areas that can be used either to save notes on their own or to modify text resources shared with other objects. They display as WINDOW objects, but brings up an editing area when the window itself is tapped. The editing area's overlays the defined window area, giving a clean, seamless editing mode. Due to their ability to be edited, EDITWINDOW objects have limited formatting when editing.

Commonly, EDITWINDOW objects are used as single-line text fields, but multiple-lines are also supported. When used in multi-line mode, the editing and display modes do not line up as well as in single-line mode.

Otherwise, EDITWINDOW objects follow the same limitations as POPEDIT objects, including the requirement that EDITWINDOWS linked to other objects must place all of its text in TRES text resources to maintain proper formatting of all objects after editing.

POPEDIT

Description:

An editable popup text window.

Required Fields:

BX BY BW BH BTEXT TEXT

Optional Fields:

PAGES STYLE FONT LINKS BFONT X Y W H COMPRESS

Supported Styles:

FILLED HORIZ_RULE SQUARE_BUTTON

Notes:

An editable text area, POPEDIT objects can be used either to save notes on their own or to modify text resources shared with other objects. The text editing window is brought up with a tap on the trigger button specified by BX,BY,BW, and BH. The POPEDIT object can be used to modify any TRES resource used by any other object, but unpredictable results may result if used to modify text used by other than WINDOW or POPWINDOW objects, as the POPEDIT control may show special control codes normally hidden from view.

SPECIAL NOTE: When the contents of a POPEDIT window can change because the POPEDIT object is linked to other objects, the contents of the POPEDIT window **must** reside in separate TRES text resource objects, or the text pre-wrapping system will get confused and mis-wrap the text after you edit one of the TEXT blocks near the top of the list of possible entries.

INPUT (New in 4.0)

Description:

An INPUT object creates a button that can modify or replace the contents of an EDITWINDOW object. Typically, they are used to create text-entry keys for forms or calculator folios.

Required Fields:

BX BY BW BH BTEXT TARGET

Optional Fields:

PAGES STYLE BFONT LINKS

Supported Styles:

FILLED SQUARE_BUTTON

Notes:

The behavior of the INPUT object is determined by the contents of the TARGET field, which takes on the following form:

TARGET   **"@<editname><mode><action><text>"**

where:

<editname>	is the **name** of the EDITWINDOW object to affect
<mode>	is either a colon ":" (**immediate mode**) or semicolon ";" (**delayed mode**).
<action>	is either a plus "+" (**add**), equals "=" (**replace**), minus "-" (**subtract**), or number sign "#" (**calculator key action**).
<text>	is the new text

In *"immediate"* mode, other screen objects which depend on the entry are immediately refreshed to reflect the new text. In *"delayed"* mode, the text is entered, but only the EDITWINDOW object itself is redrawn. Other objects such as on-screen math objects are not redrawn in delayed mode.

The *"add"* action appends text to the end of the specified EDITWINDOWs current contents, while the *"replace"* action replaces it entirely. The *"subtract"* action ignores any passed text and simply deletes the last character from the EDITWINDOW. The last action, the *"calculator"* action, acts either like *"add"* or *"replace"* action depending on whether a calculation is pending. If one has yet to be flushed out (due to entry using *delayed-mode* key presses, it adds the specified text. Once calculated, the next press replaces the contents.

Common applications include:

Calculator-style entry buttons (delayed calculation)

    TARGET "@number;#1"

    TARGET "@number;#2"

    TARGET "@number;#3"

Backspace

    TARGET "@number;-"

Enter/Calculate key (note the empty text to add)

    TARGET "@number:+"

Clear key (sets contents to nothing)

    TARGET "@number:="

Filling a form field

    TARGET "@gender:=female"

    TARGET "@gender:=male"

New to 4.20: If TARGET text is used with no target object (TARGET "a"), then the character(s) specified will be entered into the general input stream, simulating a user entering the characters by hand. If another object has the STROKE_FIND style set, it will respond to the character, allowing INPUT objects to change the current selection in an external LIST object to an entry whose first letter matches the indicated letter.

POPMERGE (New in 4.0)

Description:

Similar to POPWINDOW objects, POPMERGE objects bring up scrollable, auto-wrapping text windows when a specified trigger button is pressed.   The contents of the windows, however, can contain text from various other windows, fields, and objects, both hidden and shown. These are "merged" into the text of the final window to produce the combined output

Required Fields:

BX BY BW BH BTEXT TEXT

Optional Fields:

PAGES X Y W H STYLE FONT TABS LINKS BFONT COMPRESS

Supported Styles:

VERT_RULE HORIZ_RULE INVERTED BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER ALIGN_LEFT_START ALIGN_RIGHT_START ALIGN_CENTER_START ALIGN_LEFT_END ALIGN_RIGHT_END ALIGN_CENTER_END NO_SCROLL MASK_BUTTON_SCROLL FILLED SQUARE_BUTTON

Notes:

The text appearing in a POPMERGE popup is defined using a TEXT field in the expected way. Placeholders for text pulled from other objects are inserted using the '@' symbol followed by the name of an object to be included. Often, the source objects are hidden objects controlled by user selections. Sample text might look like:

TEXT "Madame sees your @problem difficulty will get much @prediction today!"

PRINTMERGE (New in 4.0)

Description:

A PRINTMERGE object combines text from separate objects, but instead of displaying it in a pop up window, the text is sent to TealPrint for print out on a supported printer. This allows folios to output form documents, summaries, or receipts based on user inputs and selections.

Required Fields:

BX BY BW BH BTEXT TEXT

Optional Fields:

PAGES STYLE LINKS BFONT COMPRESS

Supported Styles:

FILLED SQUARE_BUTTON

Notes:

TealPrint must also be installed on the handheld for PRINTMERGE objects to be functional.   When selected, a PRINTMERGE object will bring up a standard TealPrint printer dialog to confirm print options prior to sending the text to the printer or spooler.

MATH (New in 4.0)

Description:

A math object is a simple window displaying the numerical result of a calculation. Evaluated expressions can contain references to the current (zero-based) selection of objects with a current "value" such as lists, outlines, and tables. They can also evaluate numbers or expressions found in static objects such as windows, other math objects, and other objects which cannot be selected.

Required Fields:

X Y W H TEXT

Optional Fields:

PAGES STYLE FONT LINKS DECIMALS COMPRESS

Supported Styles:

INVERTED NO_BORDER BOLD_BORDER ALIGN_RIGHT ALIGN_CENTER

Notes:

MATH objects are one of the most complex and ambitious TealInfo objects. They show the result of a calculation as a floating-point number. To use one, a mathematical expression (equation) is specified with a TEXT field, but the result of the evaluation of the expression is what gets displayed. The number displayed can be rounded off to a number of decimal points using the DECIMALS field.

Mathematical expressions can be specified in simple, intuitive form. They are currently evaluated strictly from left to right, and in "standard" or RPN (Reverse Polish Notation) order. See the chapter "Using MATH Objects" for more information and examples on using MATH objects.

Math objects can also be used as sources for LINKs to other objects. When referenced as a LINK, only the integer portion (the part left of the decimal point) of the number is used. When referenced from an expression in another Math object, however, the full value is used.

Operands

The numbers that appear in an expression, the "operands", can simply be constant numerical values such as 1, 2, 3.1415, or .000007.

Operands can also reference another object, preceded with '@' or '#' signs. If the object is a LIST or other selectable object, the '#' (number sign) can be used to insert the current "value" of the object, which is the number of the current selection starting at zero. The '@' sign is similar, except that is inserts the numerical value of the current text, which must be either numerical or a mathematical expression itself.

If the object is a WINDOW, EDITWINDOW, other non-selectable object, or even another MATH object, then the object's entire TEXT block is evaluated as an expression and the result inserted into the current MATH object's expression. For a LIST object, only the text of the current selection is used. For instance, a simple tip calculator could include:

TEXT "(@tiprate * 0.01 + 1.0) * @subtotal"

Operators

MATH objects support a number of mathematical operators, including:

Floating Point Math

Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)

Integer Math (fractional parts of operators are ignored)
Modulus (%) – A%B is the leftover part after repeatedly subtracting B from A

Comparison (return 1 if true, 0 if false)
Greater-Than (>)
Less-Than (<)
Greater-or-Equal (>=)
Less-or-Equal (<=)
Equal (==)
Not-Equal (!=)

Logical (returns 1 if true, 0 if false)
Logical And (&&)
Logical Or (||)
Logical Xor (^^)

Bit Math
Bitwise And (&)
Bitwise Or (|)
Bitwise Xor (^)

Special
Truncate Digits (\) – truncates a number to a specified decimal place
Round Digits (:) – similar to truncate, but rounds up or down to the nearest result
Duplicate on Stack ($) – for RPN math only, duplicates the last entry on the stack

DRAW (New in 4.0)

Description:

> DRAW objects define a rectangular region whose contents are rendered according to simple graphics drawing commands. DRAW objects are typically used to draw graphs or charts in response to numerical input, or add extra graphic elements.

Required Fields:

> X Y W H TEXT

Optional Fields:

> PAGES STYLE COMPRESS LINKS CYCLE DELAY

Supported Styles:

> INVERTED BOLD_BORDER NO_BORDER

Notes:

> Powerful DRAW objects accept a series of graphics drawing commands that allow rendering of simple custom graphics or text into a folio. They can reference values in external objects, allowing them to be used to draw graphics that vary with user selections.
>
> Commands in DRAW objects are inserted in a single multi-line TEXT block of the object definition. Each line begins with a command, followed by one or more numerical or text values needed by the command. The following commands are supported:
>
> > **DLINE** x1 y1 x2 y2
> >
> > **GLINE** x1 y1 x2 y2
> >
> > **ELINE** x1 y1 x2 y2
> >
> > > Draws a single-pixel line from coordinates (x1,y1) to (x2,y2), relative to the top-left corner of the object. The (D) option draws the line in the current foreground color (default black). The (E) option erases the line to the current background color (default white). The (G) option "grays" the line drawing it in a 50% dither pattern combining foreground and background colors.
> >
> > **DRECT** x y width height corner_radius
> >
> > **GRECT** x y width height corner_radius
> >
> > **ERECT** x y width height corner_radius
> >
> > > Draws, erases, or "grays" a filled rectangle of the given dimensions. The corner of the rectangle can be set sharp or rounded using a radius of 0-4 pixels.
> >
> > **DFRAME** x y width height corner_radius thickness
> >
> > **EFRAME** x y width height corner_radius thickness
> >
> > > Draws or erases an unfilled rectangular frame (box) of the given dimensions. The radius of the corners can be set from 0-4 pixels, while the thickness of the frame can be set from 1-4 pixels.
> >
> > **DCHARS** font x y text
> >
> > **ICHARS** font x y text

ECHARS font x y text

> Draws specified text to the screen. The (D) options draws the text black-on-white, while the (I) option draws it white-on-black. The (E) options simply erases the area where text should be.

FCOLOR color

BCOLOR color

> Sets the current foreground or background colors. These options only function on a color device, and are ignored on monochrome units. They allow you to change the current foreground and background drawing colors, which default, respectively, to black and white. The colors are specified as three numerical values from 0 to 255, respectively setting the red, green, or blue component of the colors.
>
> Sample colors:
> | | |
> |---|---|
> | White | 255 255 255 |
> | Black | 0 0 0 |
> | Red | 255 0 0 |
> | Green | 0 255 0 |
> | Blue | 0 0 255 |
> | Orange | 255 200 0 |
> | Yellow | 255 255 0 |
> | Purple | 200 0 255 |
> | Gray | 150 150 150 |

To substitute values from other objects into the command list use the '@' or '#' signs followed by the name of the object to insert. The '@' sign inserts the current text from the object or the numerical equivalent of that text if a number is needed. The '#' sign returns the current "value" of the object instead, which is the zero-based number of the current selection in an outline, list, or table.   The '#' sign will reference the integer portion of a MATH object as well.

> Example: DLINE 10 10 @end_x @end_y

DRAW objects expect numerical values only, and will not evaluate expressions on their own. When math operations are needed, DRAW objects should reference hidden MATH objects to perform the calculations.

POPDRAW (New in 4.0)

Description:

> Pop-up version of a DRAW object.

Required Fields:

> BX BY BW BH BTEXT X Y W H TEXT

Optional Fields:

> PAGES STYLE COMPRESS BFONT LINIKS

Supported Styles:

> FILLED BOLD_BORDER NO_BORDER INVERTED SQUARE_BUTTON

GOTO

Description:

A button that closes the current folio and opens another one specified by name. Can also open a specified Doc file in TealDoc, movie in TealMovie.

Required Fields:

BX BY BW BH BTEXT TARGET

Optional Fields:

PAGES STYLE BFONT TARGET LINKS

Supported Styles:

FILLED SQUARE_BUTTON

Notes:

Linking to External folios:

TARGET, followed by the name of a folio in quotes specifies the folio to open when this control is tapped. Note that the TARGET must exactly match the name of the target folio *as it appears in TealInfo*, including capitalization and spacing.

Back Buttons:

Set TARGET to "$BACK" to create a button which will return back to a previous folio which may have linked to you.

Close Buttons:

A GOTO object with no TARGET field replaces the standard close button in the upper right corner of the folio.

External Programs:

TARGET can also specify the name of a TealDoc document or TealMovie movie, to launch.   Other programs may be launched as well if they support a global find operation, and have the same CreatorID of the document, but irregular behavior may occur if the other program is not specifically expecting to be called externally in this manner. With TealDoc and TealMovie, TealInfo will be re-launched after the movie or document is closed. Documents in VFS are also supported if they are in the same folder as the current folio, though the launched application must be loaded into RAM. This will almost certainly not work inf VFS with non-TealPoint applications, however.

The TARGET field functions like standard TEXT fields, in that you can specify more than one TARGET field and use the LINK field to tie the launched document to an external selection. A GOTO object with no TARGET field replaces the standard close button in the upper right corner of the folio. Otherwise, TARGET, followed by the name of a folio in quotes specifies the folio to open when this control is tapped. Note that the TARGET must exactly match the name of the target folio *as it appears in TealInfo*, including capitalization and spacing.

POPFIND (New in 4.20)

Description:

Searches for an instance of a user-entered text string in a specified LIST or WINDOW.

Required Fields:

BX BY BW BH BTEXT TARGET

Optional Fields:

PAGES STYLE BFONT LINKS

Supported Styles:

INVERTED SQUARE_BUTTON

Notes:

Specify an object to search for in the TARGET field, preceded by an '@' sign.

TARGET "@mydisplaylist"

When tapped, a POPFIND button brings up a dialog asking the user for the text to find. If the text is found in the LIST or WINDOW object specified by the TARGET field, the list or window will be scrolled to display the found text. If a list, that entry will be selected. POPFIND objects cannot find text across multiple objects or in text which is not currently loaded into a displayed object.

SETPAGE (New in 4.0)

Description:

TealInfo supports folios with up to 32 distinct pages. Displayed objects can be shown on any one or more of these pages or not at all. SetPage objects switch the current page when pressed.

Required Fields:

BX BY BW BH BTEXT

Optional Fields:

PAGES STYLE BFONT TARGET LINKS

Supported Styles:

FILLED SQUARE_BUTTON

Notes:

While all objects in a folio are present in memory and can be available for LINK operations and mathematical calculations, the SETPAGE objects determines which objects are displayed and respond to screen taps. A quoted value from 1 to 32, specified in a TARGET field, determines which page is activated when the button is tapped.

Individual objects determine which pages they appear on using the PAGES field, followed by which pages they appear on. By default, objects appear on page 1 only. If PAGES is used with no numbers, the object is hidden. SetPage objects can optionally contain PAGES fields as well.

TRES

Description:

A text resource

Required Fields:

TEXT

Optional Fields:

(none)

Supported Styles:

(none)

Notes :

TRES Text resources act as placeholder objects for holding TEXT used by other objects. They are handy to reduce file size when, say, a paragraph of text is repeatedly used in one or more objects.

TRES objects can also used to break-up TealInfo OBJECTS from exceeding the Maximum Object Text Size (See TEXT field for more info). There is about a 50-byte overhead for using a TRES object, so you won't save much space on only a few short lines of text.

TRES objects take a single field, the TEXT field. To reference a TRES object from another object, create a TEXT block which only contains the name of the TRES object preceeded by an '@' sign.

TEXT   "@resname"

TRES object can only replace an entire TEXT field. They cannot be combined or used to replace only a portion of another object's TEXT field.

See the field descriptions in the next section for more details on how to use text resources.

RANDOM (New in 4.0)

Description:

A RANDOM object provides a random value to be used in math expressions or LINKS to other objects, and a button to update its value

Required Fields:

BX BY BW BH BTEXT MAXVAL

Optional Fields:

PAGES DEFAULT BFONT

Supported Styles:

FILLED SQUARE_BUTTON

Notes:

When set, the DEFAULT field specifies the current "value" of the RANDOM object when the folio is first opened. If missing, the object will be assigned a random value from 0 to MAXVAL-1 on opening. By setting DEFAULT to MAXVAL, the Random object can start up with a value with will not be repeated when the Randomize button is tapped. This is useful to say, display an instructional prompt such as "Tap the button below to get your fortune". The PAGES field can be used with no parameters to hide the RANDOM object entirely if only the initial randomization feature is needed.

YEAR (New in 4.0)
MONTH (New in 4.0)
DAY (New in 4.0)
HOUR (New in 4.0)
MINUTE (New in 4.0)

Description:

The Year, Month, Day, Hour, and Minute objects do not have on-screen representations, and do not accept any fields or styles. Instead, they only contain a value, which can be used as LINKS to objects or values within Math objects. The values are automatically updated, allowing for the creation of special calendars or clock folios.

Notes:

All objects return values starting at 0. Values of Year objects range from 0 for the year 2000, 1 for 2001, and so on. Month objects return values from 0 (January) to 11 (December). Day objects return the day of the month minus one, ranging from 0 (day 1) to 30 (day 31). Hour objects return values from 0 to 23, minute objects from 0 to 59.

PASSWORD

Description:

The presence of this object in a folio forces causes a passkey to be entered when the folio is opened. This can be used to generate password-protected folios for security or sales-demo purposes.

Required Fields:

KEY TEXT

Optional Fields:

STYLE

Supported Styles:

LOCKOUT REGISTER

Notes:

Used for Registration

The password to be entered is defined by the KEY field. When the LOCKOUT style is defined, the folio cannot be entered until a correct key is entered. When no LOCKOUT field exists, the password request can simply be dismissed. This allows the development of "shareware" folios, using the password screen as a registration reminder.

Saving Passkey

When the REGISTER style is defined, the password is saved once it's entered if the folio is in RAM. If the REGISTER style is not set, the password is asked-for every time. You can have multiple PASSWORD objects in a folio, but this is not particularly useful.

Decompile Protection

If you provide empty TEXT ("") and **do not** set the LOCKOUT style keyword, then the passkey will not be required to read the folio, but one will need to enter it before the folio can be decompiled using the UnTIInfo program in the TealInfo Dev Kit. In this use, the PASSWORD tag should appear at the top of the list to prevent decompiling the tags above it.

User-Specific Folios

If you provide empty TEXT ("") and **do** set the LOCKOUT style keyword, then the folio will only open on a Palm whose HotSync user name matches the passkey. Use this to make secure folios that cannot be used on other Palm if copied or beamed there.

GRAFFITI

Description:

Places the graffiti shift indicator on screen. It will appear on all pages of the folio.

Required Fields:

X Y

Appendix B - MkTIInfo Field Reference

The following list details the fields used to describe the objects in Appendix A. Fields requiring a numerical value or expression such as 20+5 are followed by "(value)". Fields requiring a text string in quotes are followed by "(string)". Lastly fields requiring specialized text keywords (not in quotes) are followed by "(Keyword)".

BFONT (value)

Defines the font to be used for an object's activation button.

BTEXT (string)

Defines text for a button or trigger.

BX (value)

Defines the pixel offset of a trigger object or button from the left edge of the screen.

BY (value)

Defines the pixel offset of a trigger object or button from the top edge of the screen.

BW (value)

Defines the width of a trigger object or button.

BH (value)

Defines the height of a trigger object or button.

CYCLE (value)

This field causes WINDOW objects to cycle through their list of TEXT fields. CYCLE indicates the number of TEXT blocks to cycle through, starting with whatever field would normally be shown. There must be at least that number of TEXT fields present. For image objects, CYCLE indicates the number of images in the database to cycle through, starting with the imaged defined by RECORD.

DELAY (value)

Indicated the rate at which cycling (if turned on) occurs. This value is in tenths of seconds between frames. Thus, a value of 10 will cause cycling a one change per second.

DATABASE (string)

Defines the name of a TealPaint image database to serve as a source for imagery.

DEFAULT (value)

Defines the initial value for a list, outline, or checkmark object (starting at 0)

DECIMALS (value)

This field fixes the displayed number of digits to the right of the decimal point for results of calculations from Math objects.

DIGITS (value)

This field fixes the total displayed number of digits for results of calculations from Math objects, padding with zeroes on the left if necessary to fill the number of digits. DIGITS and DECIMALS are mutually-exclusive.

FONT (value)

Defines the font to be used for an object's body text. 0=standard font, 1=bold font, 2=large font, 3-5=standard system symbol fonts, 6=large number font. Use a program like *FontApp* (available at www.palmgear.com) to preview the correct fonts.

KEY (keyword)

Defines the password that must be entered to unlock a folio with a PASSWORD object.

LINKS (value) (value)...

The LINKS field can be used to link the contents of an object to current selection in one or more other LIST, POPLIST, OUTLINE, or CHECKMARK objects, each of which is identified by its name that follows the object TAG. An object using the LINKS tag **must follow all the objects** listed in its list of links. More information appears in the next appendix.

MAXVAL (value) ...

The MAXVAL field specifies the largest number of items a variable list can contain. Use this field to make the math work out when linking to objects whose contents can change.

PAGES (value1) (value2)…

This field determines which pages (1-32) that the current object appears. By default, all object appear on page 1 only. Specifying PAGES with no following numbers hides an object on all pages.

RECORD (value)

Used for IMAGE objects, this defines the number of the image (starting at 0) to be shown.

STYLE (keyword) (keyword)...

Followed by one or more style keywords like *ALIGN_LEFT* and *NO_BORDER*, the STYLE field determines alignment and other miscellaneous properties for screen objects.

SX (value)

For image objects, defines the horizontal offset into the source image from which the displayed image is grabbed. Must be a multiple of 8

SY (value)

For image objects, defines the vertical offset into the source image from which the displayed image is grabbed.

TABS (value)

The TABS field sets tab stop positions within an object in pixels. Up to 63 tab positions may be specified, creating 64 columns. Each tab value represents the left edge of a column measured in pixels from the left edge of the object.

If you create columns wider than the width of the object, the object will appear with a horizontal scroll bar, which can be moved to view the full width of the window, list, or outline object.

TARGET (string)

Defines the folio to open when a GOTO button is pressed, or page number for a SETPAGE object. Multiple alternate TARGET fields can be used if the GOTO object is linked to other objects. In this use, the folio opened depends on the selection of the object (typically a list) linked-to.

TEXT (string) (string)...

Defines text for this object. Text items should be quoted, with text for separate columns appearing on the same line as separate quoted groups of words. To insert a quote in a string, use two quotes together ("") inside the string.

Text for multiline text objects (such as lists and outlines) should appear as more quoted text on subsequent lines without the TEXT field name. Only TEXT fields allow information for the field to follow on subsequent lines after the field name itself.

A backslash '\' character at the very end of a line indicates that the text on the next line is a continuation of the current one. It should be used to break up a line of text but have it treated as if it were entered as a very long line of text, such as for a paragraph to be auto-wrapped by a WINDOW object. MkTIInfo has a maximum input line length of 4000 characters.

There is a maximum total text size of 32k for all TEXT blocks in a single screen OBJECT. To overcome this limit, text can be externally-referenced to TRES objects, effectively splitting up the text into one OBJECT per TEXT block.

EXAMPLE:

```
TEXT

    "This is a line of text."
    "This is another line of text, followed by a blank line."
    " "
    "This is a paragraph of text, using the backslash " \
    "character as a line-continuation marker so that " \
    "TealInfo will know to re-wrap this block as if it " \
    "were entered as a single line. Note the extra " \
    "spaces at the end of each line."
```

X (value)

Defines the object's pixel offset from the left edge (0) of the screen.

Y (value)

Defines the object's pixel offset from the top edge (0) of the screen.

W (value)

Defines the width of the object in pixels (screen is 160 pixels wide).

H (value)

Defines the height of the object in pixels (screen is 160 pixels high).

Appendix C - MkTIInfo Style Reference

The following are keywords supported by the STYLE field, where applicable.

Text Alignment

ALIGN_RIGHT

Align all text to the right of its column.

ALIGN_CENTER

Center all text in its column.

ALIGN_LEFT_START

Align the first column to the left, ignoring the global setting.

ALIGN_RIGHT_START

Align the first column to the right, ignoring the global setting.

ALIGN_CENTER_START

Align the first column in the center, ignoring the global setting.

ALIGN_LEFT_END

Align the first column to the left, ignoring the global setting.

ALIGN_RIGHT_END

Align the first column to the right, ignoring the global setting.

ALIGN_CENTER_END

Align the first column in the center, ignoring the global setting.

Appearance

INVERTED

Draw the object with reversed colors (white text on black).

FILLED

RECT: Draw it filled-in
POPLIST: Draw its button filled-in
POPEDIT: Draw its button filled-in
POPIMAGE: Draw its button filled-in
GOTO: Draw its button filled-in

SQUARE_BUTTON

Draws an activation button with square corners, not default rounded ones

HORIZ_RULE

Draw horizontal ruling lines between rows.

VERT_RULE

Draw vertical ruling lines between columns. Note that vertical rules also have a subtle effect on the behaviour of tab characters in text. Without vertical rules, tabs function as they do in a word processor, advancing the cursor to the next column to the right. If text extends past a tab stop, then next tab brings the cursor to the next tab stop to the right. When vertical rules are turned on, however, text between tabs is treated as data in a cell, truncated if necessary to fit in the available space before the next tab stop, so data never overflows into a neighboring column.

NO_BORDER

Don't draw a border around the object.

BOLD_BORDER

Draw a thicker border around the object.

ROUND_BORDER

Draw a rounded border around the object.

User Interface

NO_SCROLL

Don't allow a scroll bar to appear and don't reserve pixels for one when word wrapping.

STROKE_FIND

Move the current selection to the next line whose first char matches an entered stroke.

BUTTON_SCROLL

Do page-by-page scrolling of a text object when scroll buttons are pressed. Default settings scrolls vertically with the Page Up/Down scrolling buttons. Using variations on the BUTTON_SCROLL keyword, it's possible to map combinations of the four applications buttons (Datebook, Address, To Do List, and Memopad) to vertical and/or horizontal scrolling for objects.

When mapped to a folio in this way, the default behavior of the application buttons used is temporarily overridden only while the folio is open, and for POPIMAGE, POPTEXT, and POPLIST objects, only when the popup is shown.

When overriding such buttons, it's important to indicate the new button mapping in the folio to avoid confusing the user.

Other Supported Variations include:

BUTTON_SCROLL_A

Vertically scroll with Page Up/Down buttons

BUTTON_SCROLL_B

Vertically scroll with Date/Addr buttons

BUTTON_SCROLL_C

Vertically scroll with Addr/ToDo buttons

BUTTON_SCROLL_D

Vertically scroll with ToDo/Memo buttons

BUTTON_SCROLL_E

Horizontally scroll with Page Up/Down buttons

BUTTON_SCROLL_F

Horizontally scroll with Date/Addr buttons

BUTTON_SCROLL_G

Horizontally scroll with Addr/ToDo buttons

BUTTON_SCROLL_H

Horizontally scroll with ToDo/Memo buttons

BUTTON_SCROLL_I

Horizontally scroll with Date/Memo buttons

BUTTON_SCROLL_J

Vertically Page Up/Down, Horizontally Date/Addr

BUTTON_SCROLL_K

Vertically Page Up/Down, Horizontally Addr/ToDo

BUTTON_SCROLL_L

Vertically Page Up/Down, Horizontally ToDo/Memo

BUTTON_SCROLL_M

Vertically Date/Addr, Horizontally ToDo/Memo

BUTTON_SCROLL_N

Vertically ToDo/Memo, Horizontally Date/Addr

BUTTON_SCROLL_O

Vertically Addr/ToDo, Horizontally Date/Memo

BUTTON_SELECT

Move the current selection up or down when scroll buttons are pressed. Default settings move the current select with the Page Up/Down scrolling buttons. Using variations on the BUTTON_SELECT keyword, it's possible to map combinations of the four applications buttons (Date, Address, To Do, and Memo) to scrolling movement as well. When mapping a single button to a POPLIST object, the object acts like the category selector in the standard

application, cycling through available choices. As with the BUTTON_SCROLL variations, when overriding the application buttons, it's important to indicate the new button mapping in the folio to avoid confusing the user.

Other Supported Variations include:

BUTTON_SELECT_A

Change selection up/down with the Page Up/Down buttons

BUTTON_SELECT_B

Change selection up/down with the Date/Addr buttons

BUTTON_SELECT_C

Change selection up/down with the ToDo/Memo buttons

BUTTON_SELECT_D

Change selection down (looping) with the Date button

BUTTON_SELECT_E

Change selection down (looping) with the Addr button

BUTTON_SELECT_F

Change selection down (looping) with the ToDo button

BUTTON_SELECT_G

Change selection down (looping) with the Memo button

Special Styles

COMPRESS

The COMPRESS style forces an entire object to be stored compressed in the folio. Compression typically results in a 35-40% reduction in the size of the object. As temporary storage and decompression time is needed to work with compressed objects, compression should be used carefully to avoid performance or memory problems. In general, do not use compression for objects referenced by MATH expressions or that need to be drawn often. It is generally safe to compress popup objects, as they are used one at a time and are not accessed when they are not popped up.

LOCKOUT

PASSWORD: Require the key to be correct to enter the folio.

REGISTER

PASSWORD: Save a registration key once entered.

Appendix D - Credits

Manual by Vince Lee, Tex Tennison, Sara Houseman, and Diane Dybalski

Appendix E - Contact Info

TealInfo by TealPoint Software
©1999-2002 All Rights Reserved.

TealPoint Software
454 Las Gallinas Ave #318
San Rafael, CA 94903-3618
We look forward to hearing from you.

Please visit us at www.tealpoint.com, or email us at contact@tealpoint.com.

Appendix F - Disclaimer

We at TealPoint Software are committed to providing quality, easy-to-use software. However, this product is provided without warranty and the user accepts full responsibility for any damages, consequential or otherwise, resulting from its use.

This archive is freely redistributable, provided it is made available only in its complete, unmodified form with no additional files and for noncommercial purposes only. Any other use must have prior written authorization from TealPoint Software.

Unauthorized commercial use includes, but is not limited to:
- A product for sale.
- Accompanying a product for sale.
- Accompanying a magazine, book or other publication for sale.
- Distribution with "Media", "Copying" or other incidental costs.
- Available for download with access or download fees.

This program may be used on a trial basis for 30 days. The program will continue to function afterwards. However, if after this time you wish to continue using it, please register with us for the nominal fee listed in the program.

Thank you.