The M Windowing API: The Tools

by Gardner S. Trask III

This is the second article of a three-part series on the M windowing application program interface (MWAPI). The first article, published in the February 1994 issue of *M Computing*, introduced the realities of how the move to windows will affect M's programming methodology, style, and code. This second article introduces M professionals to the tools or palette of the MWAPI. The third article will create an application.

A Rose by Any Other Name

As with many aspects of the computer world, there is little universal agreement on terms and definitions in the interface area. Different schools of technology describe the same object in many ways, users twist the terms for comfort or humor, and vendors each try to individualize and self-promote by adding custom names to an object or process. Where does this leave a new user? The new user is left to learn by assimilation, a greatly hampered process.

Intentionally, the MWAPI was designed to conform to no one window interface, platform, or architecture. By encompassing all windows platforms, MWAPI authors have defined terminology that, while using generally accepted terms, are platform-independent. That which the X-Motif user calls a gadget and that which an MS-Windows user calls a gadget is of no concern. That which MWAPI calls a gadget is important here.

Some Assembly Required

Putting together the MWAPI components is an exciting prospect. It is not unlike getting a model-building kit as a kid. Did you really want to first separate and inspect each labeled airplane part to ensure all pieces were enclosed? You probably wanted to jump right in and begin glueing the pieces together, skipping over the inspection step. In order to start building the MWAPI, however, I will first label all the pieces in the package. (Keeping an inventory is also a good idea.) Building the MWAPI application will be the subject of the third article, as noted in the introductory paragraph.

Of necessity, this article foregoes a complete discussion of each definition. An excellent reference for in-depth technical information about the MWAPI is the MUMPS Development Committee (MDC) document X11/SC11/TG4/WG6.

Windows, Elements, and Events

The graphical user interface (GUI) of MWAPI consists of one or more screens (windows) on which various elements (gadgets, menus, timers) are manipulated by various events. These windows are placed on display, and the events are monitored by an event driver.

The MWAPI runs as M code within a vendor's system (such as Micronetics or InterSystems' DataTree) while running on top of a windows platform (this may be MS-Windows, X-Motif, etc.). This would be known as MSM or DTM running on MS-Windows or X-Motif. The actual windows platform is irrelevant to the MWAPI.

Each M process has a ^$DISPLAY allowing a place to create one or more ^$WINDOWs with one or more elements. Elements are gadgets, menus, and timers. Gadgets are objects for entering, modifying, and or displaying data or graphic representations. Menus allow the selection of processes. Timers generate events in a timed manner. There are mechanisms to control events at the job level, window level, and each element level.

```
^$DISPLAY
                      ^$EVENT - Process event-handling
     |
     ^$WINDOW 1             (window event-handling)
          |                 (gadget event-handling)
          gadgets
          menus             (menu event-handling)
          timers            (timer event-handling)
     |
     ^$WINDOW n ...
     .
     .
     .
```

Figure 1. A representation of an ssvn hierarchy wherein ^$DISPLAY can have one or more ^$WINDOWS. ^$WINDOW can have some number of gadgets, menus, and/or timers. It also illustrates the many levels of optional event-handling.

This window and element creation and manipulation is accomplished through the use of a powerful language feature, unique to M Technology, called structured system variables (ssvns). ssvns are variables (with a global array-like design) that have a predetermined structure, and in some cases, value. This provides a vendor-independent mechanism for retrieving information or manipulating the system. Commonly, ssvns store in a now standard location that which was previously vendor-specific information. For example, the proposed standard calls for ssvn ^$LOCK to return LOCK table information regardless of vendor, thus eliminating many Z functions and even MGR tools.

In the context of windows, the MWAPI ssvns are unique for each job or process. This means the user may set attributes to the main window and not interfere with a coworker's main window. The MWAPI ssvns also have the ability to be SET and KILLed, thus allowing an application to manipulate them "on the fly."

The three ssvns associated with the MWAPI are ^$DISPLAY, ^$EVENT, and ^$WINDOW. A description of each follows.

^$DISPLAY

The ^$DISPLAY (^$DI) ssvn gives the MWAPI a logical surface for displaying input and output via windows. Normally one would consider the monitor as an ^$DISPLAY, but ^$DI can span more than one physical device (such as a bank of monitors or remote site screens), or can exist as a portion of a physical device (such as a multiprocess workstation monitor). Each M process maintains its own $DISPLAY (or possibly multiple ^$DIs) and is not shared by other processes. The system variable, $PDISPLAY, specifies the principal display device. Depending on the underlying windows platform, the physical device, and the vendor's implementation, a user can control many characteristics of the ^$DISPLAY. (See figure 2.)

^$EVENT

The ^$EVENT ssvn (^$E) will set and retrieve information about events, and is accessible to a process during callbacks for the event.

Used in conjunction with ^$WINDOW, the ^$E ssvn contains information about the most recent event that occurred. This could include the last keys pressed or the next window that gets focus. The ^$E ssvn can be used to control the flow of processing. ^$E controls the events for the current process. Other mechanisms control events for the windows and each element (gadgets, timers, and menus).

^$E may contain some or all of the characteristics in figure 3.

^$DISPLAY(displayname,	
"BCOLOR")	Default background color
"CLIPBOARD")	Contents of the platform's clipboard
"COLOR")	Window application area color
"COLORTYPE")	Grey scale, mono, or color capabilities
"FCOLOR")	Gadget default foreground color
"FOCUS")	Window (and perhaps gadget) with focus (0: no; 1: yes)
"KEYBOARD")	Does display have a keyboard?
"PEN")	Does the display have a pen I/O device?
"PLATFORM")	Name and version number of the platform
"PTR")	Does the display have a pointer, e.g., a mouse?
"SIZE")	Horizontal, vertical size of display in units?
"SPECTRUM")	Number of color or grey scales available
"TYPEFACE")	Specifies the fonts, faces, and point sizes available
"UNITS")	Units used in sizing, e.g., char, pixel, etc.

Figure 2. ^$DI characteristics.

^$EVENT(	
"CHOICE")	The 'choice' of a SELECT event, e.g., which radio button
"CLASS")	Specifies the event class to which the event belongs
"ELEMENT")	Specifies the element to which the event belongs
"KEY")	Identifies the key or keys pressed for keyboard event
"NEXTFOCUS")	Focus on unfocus or change event object that will next gain focus
"OK")	Specifies if the event should be compiled normally
"PBUTTON")	Identifies button associated with pointer event
"PPOS")	Pointer position
"PRIORFOCUS")	Past unfocus or change event object that had focus
"PSTATE")	Pointer buttons (other than PBUTTON) during a pointer event
"SEQUENCE")	Unique sequence number of event within process
"TYPE")	Identifies the type of event, e.g., pointer, select
"WINDOW")	The name of the window for which the event occurred

Figure 3. ^$EVENT characteristics.

^$WINDOW

The heart of the MWAPI is ^$WINDOW (^$W). ^$W is the ssvn that allows users to create, modify, and delete windows. It is a place to create, modify, and delete the gadgets, timers, and menus accessible to a particular process. There are multiple levels to the ^$WINDOW ssvn. There is a level to control attributes of the window itself, for window events, for gadgets, for menus, and for timers. At the base level, each window has certain attributes, such as those shown in figure 4.

^$WINDOW(windowname,

"ACTIVE")	Indicates if window, elements, and descendants are active
"BCOLOR")	Default background color for subsequently created gadgets
"COLOR")	Color of application area
"DEFBUTTON")	Specifies default push button for a window
"DISPLAY")	Specifies the display where window appears
"FCOLOR")	Default foreground color for subsequently created gadgets of this window
"FFACE")	Default typeface for subsequently created gadgets of this window
"FSIZE")	Default font size for subsequently created gadgets of this window
"FSTYLE")	Default font style for subsequently created gadgets of this window
"ICON")	Specifies icon to use if iconified
"ICONIFY")	Switch to allow window to be iconified
"ID")	Internal identification number of window
"ITITLE")	Icon title
"MENUBAR")	Identifies the menu to be displayed at the window level
"MIN")	Identifies if this window currently exists as an icon
"MODAL")	Specifies modal window and type of modality. Modal disables various other windows, depending on type
"NEXTG")	Defines action when window, but no gadget, gets focus
"PARENT")	Defines the window's parent. Used to inherit attributes
"POS")	Defines original position of window frame, in units
"RESIZE")	Indicates if the user can resize the window
"SCROLL")	Indicates a horizontal or vertical scroll bar is present
"SIZE")	Specifies the window viewport height and width, in units
"SIZEMIN")	Specifies the minimum size allowed to resize the window
"SIZEWIN")	Specifies the window frame height and width, in units
"TIED")	Indicates if window position is relative to parent window
"TITLE")	Specifies the window title text
"TYPE")	Specifies window type
"UNITS")	Specifies measurement units, e.g., pixels, characters
"VISIBLE")	Indicates if a window is visible to users

Figure 4. Window attributes.

Note that while there appear to be many attributes at each level, the power of inheritance allows default values of the underlying windows platforms, and any parent windows, to predefine most of these attributes. The user thus can create complex windows by modifying a relatively small subset of attributes. For example, the default background color (BCOLOR) for a gadget's window is passed by the windowing platform, and the default font size (FSIZE) is 12. Unless the developer has a specific reason to change these defaults, they are automatic and need not be set explicitly, ever.

This inheritance property holds for almost all the display, event, window, gadget, menu, and timer attributes. For example, to create a simple window with a title of "Hello World!" (see figure 5), all the programmer needs to do is SET ^WINDOW(1,"TITLE")="Hello World!".

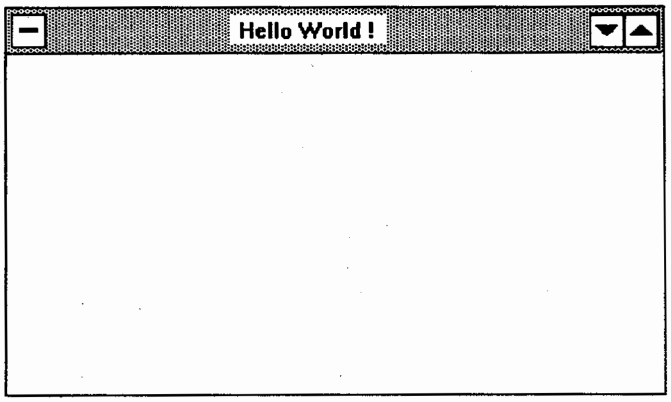

Figure 5. A simple window requires only a simple command.

The inheritance property will generate automatically the global structure as shown in figure 6.

```
^$WINDOW(1,"DISPLAY")="1"
^$WINDOW(1,"FCOLOR")="0,0,0"
^$WINDOW(1,"FFACE")="System"
^$WINDOW(1,"FSIZE")="10"
^$WINDOW(1,"FSTYLE")="BOLD"
^$WINDOW(1,"ICON")=""
^$WINDOW(1,"ICONIFY")="1"
^$WINDOW(1,"ID")="5070"
^$WINDOW(1,"ITITLE")="Hello World!"
^$WINDOW(1,"MIN")="0"
^$WINDOW(1,"POS")="66,66,PIXEL"
^$WINDOW(1,"RESIZE")="1"
^$WINDOW(1,"SCROLL")="0"
^$WINDOW(1,"SIZE")="236,42,PIXEL"
^$WINDOW(1,"SIZEMIN")=""
^$WINDOW(1,"SIZEWIN")="244,69,PIXEL"
^$WINDOW(1,"TIED")="0"
^$WINDOW(1,"TITLE")="Hello World!"
^$WINDOW(1,"TYPE")="APPLICATION"
^$WINDOW(1,"UNITS")="PIXEL"
^$WINDOW(1,"VISIBLE")="1"
```

Figure 6. Example of one M command generating a global structure.

Window Events

As with the process event ^$E, each window can respond to events in various ways. Default event-handling can be passed to the window, but as always, the programmer can modify these. The ^$W events are shown in figure 7.

And Now, the Rest of the Story

Figure 8 displays most of the gadgets available through the MWAPI as well as a menu bar. (Timers are not visible objects, so there is no way to represent them visually.)

But first, the Timer, which has no visual representation. A Timer generates a specified event after a specified time interval. Usually used for time-out events, the timer structure is very simple:

```
^$W(windowname,"T",timername,
"INTERVAL")             Number of seconds for event to
                        take place
"ACTIVE")               0 - timer inactive, 1 - timer
                        active
"EVENT","TIMER")        Routine to call if timer expires.
```

Follow along with the figure, which shows each of these features.

```
^$WINDOW(windowname,"EVENT",

                Process to invoke when
"CLICK")        there is a single click of
                pointer device
"CLOSE")        Request to "destroy" window
"DBLCLICK")     There is a doubleclick of
                pointer device
"FKEYDOWN")     Function key is pressed
"FKEYUP")       Function key is released
"FOCUS")        Window receives focus
"HELP")         Call for help
"KEYDOWN")      Key is pressed
"KEYUP")        Key is released
"MAX")          Window is expanded to maximum
                size
"MIN")          Window is minimized
"MOVE")         Window is moved
"PDOWN")        A pointer down event occurs
"PDRAG")        A pointer drag event occurs
"PMOVE")        A pointer move event occurs
"PUP")          A pointer up event occurs
"RESIZE")       A resize event for the window
                occurs
"UNFOCUS")      A window loses focus
```

Figure 7. Window events that a programmer can modify.

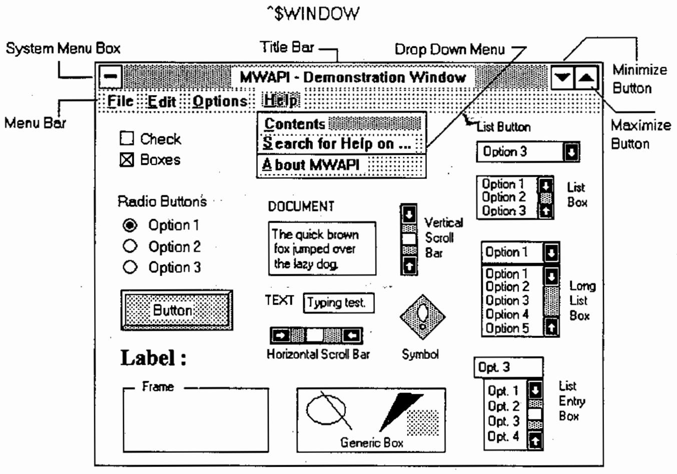

Figure 8. The MWAPI demonstration window.

Menus, such as that in the figure, allow users to select from a list of choices. Menus appear as menu bars, which are shown (horizontally) directly below the window's title bar, or as drop-down menus (or pop-up) that display options vertically below the menu bar when that is displayed at the top. Drop-down menus can invoke other drop-downs. The structure of a menu would include such attributes as ACTIVE, ID, POS, and substructures to identify each CHOICE and associated events. Refer to user manuals and/or MDC documents for further detail.

Now for the most exciting, most visible, element: the gadget. Gadgets allow interaction within windows to create, retrieve, and modify data or invoke processes. Obviously, gadgets in figure 8 provide a more intuitive and visually eye-pleasing mechanism for data manipulation than the old roll-and-scroll format, or even than formatted screens on character-node displays. The defined MWAPI gadgets appear in the figure, and are as follows:

Check Box
: Allows the user to turn an indicator on or off.

Document
: Enables the user to view, enter, and modify one or more lines of text.

Generic Box
: Allows the user an area in which to present, enter, or modify text, geometric figures, and other graphical figures.

Group Frame
: A rectangular outline used to visually group gadgets together on a window. Example: a frame encompassing several check boxes similar in nature.

Label
: Static text that appears within a window.

List Box
: Displays a list of items and allows the user to select one or more. Should the list exceed the box size, a scroll bar can enable the user to move through the text.

List Button
: Displays a text line and a push button. The text line is filled with the default or selected choice. When users push the button they receive the (normally invisible) list of available items. When the user selects one item, the list box disappears, leaving the selection in the text box. The choices cannot be edited by the user.

List Entry Box
: Similar to the list button, the list entry box gives the user a text line and a push button. It also displays a defined number of options in a list box automatically. The user navigates lists exceeding the list box size via scroll bars. When the user selects one item, the text-entry line is filled with selections. The user also could enter new items on the text-entry line.

Long List Box
: Similar to a list box, the long list box helps the user manipulate long lists of items. The long list box has special events to allow the application to control scrolling, navigation, and display of list items. Multiple items can be selected.

Push Button
: A rectangular button with associated text, that when "pushed," generates an event.

Radio Button Set
: A collection of related items that a user can select. At any time, only one item can be selected. Selecting an item "unselects" any previously selected item.

Scroll Bar
: A scroll bar is a visual representation of values on a numeric scale. These can be horizontal or vertical. The user can change the value by moving the position indicator.

Symbol
: A symbol is an image to display within a window. Symbols are for viewing purposes only and cannot get focus or be modified.

Text
: A text box that allows the user to view, enter, and modify a single line of text. If the text line cannot fit within the box, a horizontal scroll bar will aid in navigation.

Gadget Attributes

The following attributes are similar in scope to those for the window:

```
^$W(windowname,"G",gadgetname,
```

ACTIVE, BCOLOR, FCOLOR, FFACE, FSIZE, FSTYLE, ID, NEXTG, POS, SCROLL, SIZE, TITLE, TYPE, UNITS, VISIBLE

The attributes listed in figure 9 are also associated with gadgets.

"CANCEL")	Do not perform focus, change, or unfocus events
"CANCHANGE")	Users are allowed to change values
"CHANGED")	Flag is set if the gadget value changes
"CHARMAX")	Maximum characters of text
"CHOICE")	Choices for radio buttons, list boxes, etc.
"DRAW")	Defines the draw commands that are descendants
"DRAWTYPE")	Defines type of draw commands
"EVENT")	Dependent events that can happen to gadget
"FRAMED")	Does frame appear around gadget?
"INSELECT")	Insertion point of new text
"NEXTG")	Defines next gadget to get focus
"RESOURCE")	Specifies an image to display
"ROWCOL")	Specifies how choices are displayed on a RADIO gadget
"SCROLLBY")	Scroll increment when user presses movement button
"SCROLLDIR")	Horizontal or vertical scroll bar
"SCROLLPOS")	Position of indicator in long list box
"SCROLLRANGE")	Value range of scroll bar
"SELECTMAX")	Maximum concurrent selection allowed
"SELECTVAL")	Contains the selected data
"TBCOLOR")	Title background color
"TFCOLOR")	Title foreground color
"TFFACE")	Title font face
"TFSIZE")	Title font size
"TFSTYLE")	Title font style
"TOPSHOW")	List value in top line of gadget
"TPOS")	Title position

Figure 9. Gadget attributes.

The following is a list of event types that gadgets can act on. They exist in the structure:

```
^$W(windowname,"G",gadgetname,"EVENT",eventtype
```

These are similar to events for windows:

CLICK, DBLCLICK, FKEYDOWN, FKEYUP, FOCUS, HELP, KEYDOWN, KEYUP, PDOWN, PDRAG, PMOVE, PUP, UNFOCUS

The event types shown in figure 10 are unique to gadgets.

```
"CHANGE")        Event to complete if gadgets
                 "CHANGED" attribute is set
"GOBOTTOM")      Go to bottom movement control
                 is selected
"GODOWN")        Go down movement control is
                 selected
"GODOWNBIG")     Go down big movement control is
                 selected
"GOTOP")         Go to top movement control is
                 selected
"GOUP")          Go up movement control is
                 selected
"GOUPBIG")       Go up big movement control is
                 selected
"SELECT")        Event if user selects or
                 deselects a gadget or choice
```

Figure 10. Gadget event types.

Conclusion

Such are the building blocks of the MWAPI. In this article, I have given you all the "model airplane set" pieces and explained them. The next article will demonstrate how to put them together.

It will discuss not only the creation of an application, but also how to manipulate these ssvns, how Merge and Set differ, the side effects of MERGE, inheritance, focus, unfocus, modal windows, estart/estop, and much, much more.

Endnotes

1. G. Gardner, "Peeking at the New M Windowing API," *M Computing* 1:2 (April 1993).
2. Microsoft Programming Series, *The Windows Interface: An Application Design Guide* (Penguin Books, 1992).
3. MDC Subcommittee 11, M Windowing API, X11/SC11/TG4/WG6/93–17, 29–JUNE–1993.
4. Micronetics, *MSM-GUI Reference Manual* (1993).
5. Digital Equipment Corporation, *Windowing Application Programming Interface Guide for DSM* (December 1993).

The following table shows which attributes are available for which gadgets:

A - Check Box	B - Document	C - Generic Box
D - Group Frame	E - Label	F - List Box
G - List Button	H - List Entry Box	I - Long List Box
J - Push Button	K - Radio Button	L - Scroll
M - Symbol	N - Text Box	

Elements and Their Attributes														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
ACTIVE	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓		✓
BCOLOR	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓
CANCEL	✓	✓	✓			✓	✓	✓	✓	✓	✓			✓
CANCHANGE		✓				✓	✓	✓	✓					✓
CHANGED	✓	✓				✓	✓	✓	✓		✓	✓		✓
CHARMAX		✓						✓						✓
CHOICE						✓	✓	✓	✓		✓			
DRAW			✓											
DRAWTYPE			✓											
EVENT	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓		✓
FCOLOR	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓
FFACE		✓	✓			✓	✓	✓	✓		✓			✓
FRAMED		✓	✓		✓						✓			✓
FSIZE		✓	✓			✓	✓	✓	✓		✓			✓
FSTYLE		✓	✓			✓	✓	✓	✓		✓			✓
ID	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
INSELECT		✓						✓						✓
INTERVAL														
NEXTG	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
POS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RESOURCE										✓			✓	
ROWCOL											✓			
SCROLL		✓												
SCROLLBY												✓		
SCROLLDIR												✓		
SCROLLPOS									✓					
SCROLLRNGE									✓			✓		
SELECTMAX						✓			✓					
SELECTVAL		✓						✓						✓
SIZE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
TBCOLOR		✓				✓	✓	✓	✓		✓			✓
TFCOLOR		✓				✓	✓	✓	✓		✓			✓
TFFACE	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
TFSIZE	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
TFSTYLE	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
TITLE	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
TOPSHOW						✓		✓	✓					
TPOS		✓				✓	✓	✓	✓		✓			✓
TYPE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
UNITS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VALUE	✓	✓				✓	✓	✓	✓		✓	✓		✓
VISIBLE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Gardner S. Trask III is a senior programmer/analyst for Collaborative Medical Systems in Waltham, Massachusetts, and owner of MPower Computer Consultants in Beverly, Massachusetts. He is the ex-officio chairperson of the New England M Users' Group. As the system operator of the NEMUG Bulletin Board System, he can be reached via the NEMUG BBS [Phone: (508) 921-6681 (8-N-1)] or at trask@world.std.com (via Internet). This article may or may not reflect the views of Mr. Trask's employers.