MWAPI: A New Face for M

by James Hay

Portability

Although this particular detail has not been mentioned often in the press lately, there is a M[UMPS]-related standard for dealing with windows. When this standard was established about six years ago, it was intended to approach window programming "from a higher level perspective," so that programs could use the same application-level code, regardless of the actual windowing platform on which the programs were to be executed. In those days, the popular platforms were Apple Macintosh™ and X-Window™ and Microsoft Windows 3.1™, and for each of these platforms, at least one vendor demonstrated an implementation.

At the time, Microsoft Windows 3.1 was not really seen as a serious contender in the world of windows. As the world changed, the Microsoft products have been touted as the "winner" in the desk-top arena. Now that Linux and "Open Source" environments are gaining popularity, the world is changing again, and Microsoft is no longer seen as the "sole target environment."

As a result, we will probably see a renewed interest in the MWAPI standard (ANSI/MDC X11.6-1995), as an increasing number of software authors will want to write code that is portable between Microsoft's windowing environments and the X-Window™ environments in the Linux-based world.

—Ed de Moel

The MUMPS Windowing Applications Programmer Interface (MWAPI) extends the M language into the area of windows development. Programming in a graphical environment requires change from the methodologies used in the traditional character cell technology of standard M. This article takes a look at the similarities and differences the M programmer will encounter developing applications with the MWAPI.

The MWAPI specification adds three new Structured System Variable Names (SSVNs), a few string functions and a couple of special variables. The new SSVNs ^$Display, ^$Window and ^$Event hold the definitions of GUI entities used in the MWAPI system. The string functions and special variables make working with MWAPI entities, structures, and graphical environments easier for the programmer.

Developers of MWAPI applications enjoy the same high-level platform-independent development environment and portability of source code as programmers of standard M applications. The high-level approach removes the developer from the intricacies of any particular windowing environment while maintaining the look-and-feel characteristics determined by the host where the application will be used. A common set of syntax and methods is abstracted from the operating windowing environment so MWAPI-created GUI's can take on these unique characteristics, from disparate windowing environments, with no change to source code. The abstraction layer is transparent to the user - a behind-the-scenes approach. This provision permits the developer to focus attention where it is most appropriate - the application.

The most advantageous use of the MWAPI is the development of GUI applications, although support is provided for character cell front ends. Legacy applications are supported through a terminal emulation window, MTERM, similar to a "DOS Box." An MTERM window is considered as a device where input and output operations can be redirected. This device definition permits unaltered legacy applications to be used in a graphical windowing environment.

Windows, gadgets, menus, and timers make up MWAPI GUI interfaces, and are each defined by a set of attributes. The MWAPI defines a list of keywords to describe the attributes that define the makeup of the windows, gadgets, menus and timers. Quality or quantity is assigned to attributes nodes to define properties of GUI components. For example

the attribute BCOLOR describes a background color attribute, while a value of 65545,0,49150 defines the color quality. A UNITS value of 10 assigns 10 units in the unit-of-measurement. Units of measurement can be: character, pixel, point, relative and implementation-specific.

Windows and elements are created by simply assigning values to their attributes in the ^$Window SSVN. Example One shows the syntax for assigning a value to a window and Example Two provides the same for one of its gadgets.

Example One

```
^$W(window name, attribute keyword)=value
```

Example Two

```
^$W(window name,"G",gadget name,attribute)=
            value
```

The same commands used to assign values to variables in standard M applications are used to assign values to MWAPI-defined SSVNs. The creation of MWAPI entities and elements is a side effect of the assignment of values to their attributes. Using the syntax shown in Example one as the argument to a Set command, a window named "MAIN" with the title "ACCOUNTS" is created by the assignment shown in Example Three.

Example Three

```
SET ^$W("MAIN","TITLE")="ACCOUNTS"
```

The set command works for creating windows and for modifying attribute values of entities and elements already created. The size, position and type of gadget must be defined before their attribute keywords and values can be assigned to the ^$Window SSVN. The Merge command must be used to create a gadget. The Merge command assigns the array nodes and values of the attribute keywords into the ^Window SSVN to create the gadget. Example Four creates a scroll gadget with the name "SCROLL1" 20 by 25 units in the unit of measurement at position 80,40 in window "MAIN."

Example Four

```
SET W("MAIN","G","SCROLL1","TYPE")="SCROLL"
SET W("MAIN","G","SCROLL1","POS")="80,40"
SET W("MAIN","G","SCROLL1","SIZE")="20,25"
MERGE ^$W=W
```

Creating an element can have the effect of creating both its parent window and the abstraction layer between the interface and the windowing environment. For example: If the window in Example Four ("MAIN") is undefined when the assignment is made to create an element for it (a menu or timer can be used instead of the gadget assignment), the window will be created. If the abstraction layer is not defined, it too will be created. This property of MWAPI development provides the programmer with the means to rapidly create an interface.

Default values are implicitly assigned to define the attributes inherent in a window and element that were not defined through explicit statements. The entities and elements created by Examples Three and Four do not include statements to define all the inherent characteristics of each. Default values fill in the gaps to define attributes not stated. Default values are passed down from parent to child, as is found in a family lineage. Children share characteristics with their parents, and this is true with an MWAPI GUI. Each M Process has its own windows interface, which has the tree structure shown in Illustration 1.

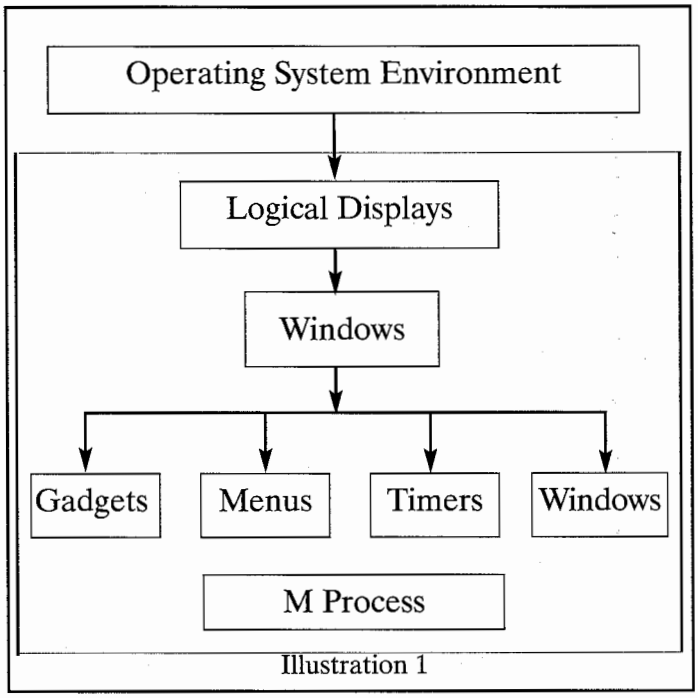

Illustration 1

The illustration shows that within an M process logical displays are descended from the windowing platform where the interface is operating. Multiple logical displays can exist to define abstraction layers for multiple and separate interface environments. Multiple windows can descend from each logical display, and each window can have as their children numerous gadgets, menus, timers and even other windows. The operating system environment is placed outside the interface hierarchy to show

```
SCROLL ; this tag processes SELECT events for gadget SCROLL1
 DO WINELMT ; get event information
 SET X=^$W("MAIN","G",ELEMENT,"VALUE") ; get the gadget's value
 SET ^$W("MAIN","G","NUMBER","VALUE")=X ; assign the value to gadget NUMBER
 QUIT

WINELMT ; get characteristics of an event
 SET SEQUENCE=^$E("SEQUENCE") ; get the sequence of the event
 SET WINDOW=^$E("WINDOW") ;get the window's name
 SET X=$G(^$E("ELEMENT"))
 SET ELTYPE=$P(X,",",1) ; Get the element's type.
 SET ELEMENT=$P(X,",",2) ; Get the element's name.
 SET TYPE=^$E("TYPE") ; get the event type
 QUIT
```

Example 6

that it may be omitted from the functioning of an MWAPI application.

The server applications to an MWAPI interface client "hand shake" with each other through an event queue. The high level design of the MWAPI uses the underlying windowing platform's event queue while maintaining the platform independence of the source code. The MWAPI interprets events that transpire in windows and elements as a characteristic of the window or element. The same methods used previously to define windows and elements are used to define actions when an event occurs in a window or element. In the ^$Window SSVN, the level below the attribute name, EVENT, is a keyword that describes the event. The value assigned to this level is the argument to a Do command where processing is directed on the server application, when the event occurs. Since gadget SCROLL1 already exists, the Set command can be used to add the attribute to it. Example Five shows the assignment of an event node for the gadget created in Example Four.

Example Five

```
SET ^$W("MAIN","G","SCROLL1","EVENT",
"SELECT")="SCROLL^CHECKING"
```

If the processing logic called from an event node in the ^Window SSVN needs to handle an event that occurred for the calling entity, it will need to reference the event object. The event object is normally redefined for each MWAPI event where an event node exists with an event keyword that describes the event. The event object is a list of event information attributes and values assigned to the ^Event SSVN. The design of the application used

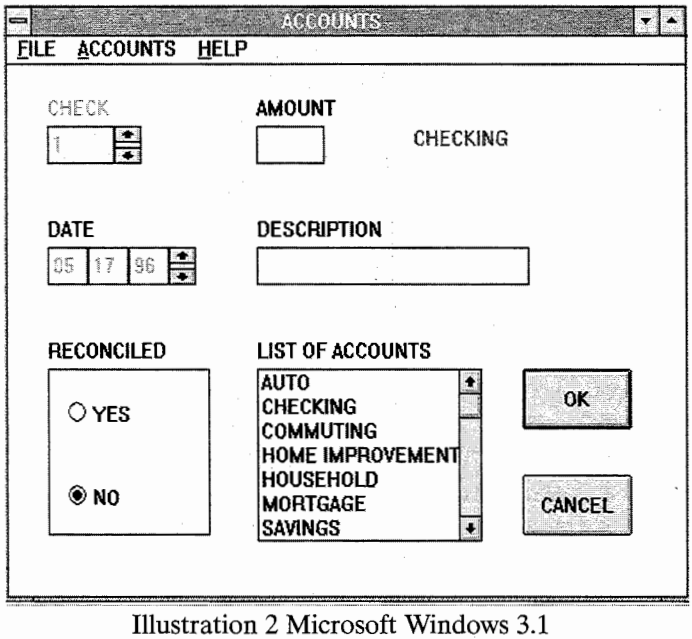

Illustration 2 Microsoft Windows 3.1

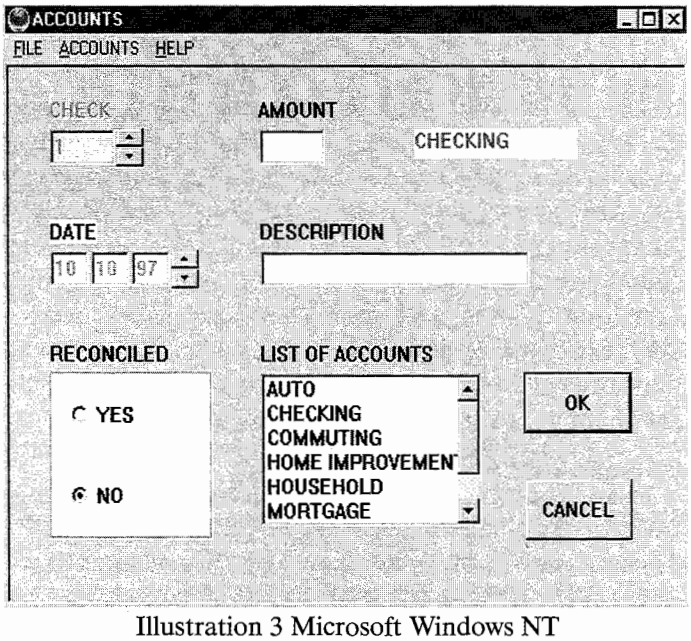

Illustration 3 Microsoft Windows NT

```
EN
;Other windows may be in use, so kill only those used by this application.
 NEW W,MWDATE,MWMTH,MWDAY,MWYR,MWNUM,MWTIME,MWFFACE,MWFSIZE,MWFSTYLE
 NEW MWUNITS,X,Y
 KILL ^$W("MAIN")
 SET W("MAIN","TITLE")="ACCOUNTS"
 SET W("MAIN","POS")="40,40"
 SET W("MAIN","SIZE")="500,350"
 SET W("MAIN","MENUBAR")="MENUBAR"
 SET W("MAIN","NEXTG")="NUMBER"
 SET W("MAIN","DEFBUTTON")="OK"
 SET W("MAIN","EVENT","CLOSE")="CLOSE^CHECKING"
 ; menus
 SET W("MAIN","M","MENUBAR","CHOICE",1)="&FILE"
 SET W("MAIN","M","MENUBAR","CHOICE",1,"SUBMENU")="FILE"
 SET W("MAIN","M","MENUBAR","CHOICE",2)="&ACCOUNTS"
 SET W("MAIN","M","MENUBAR","CHOICE",2,"SUBMENU")="ACCOUNTS"
 SET W("MAIN","M","MENUBAR","CHOICE",3)="&HELP"
 SET W("MAIN","M","MENUBAR","CHOICE",3,"EVENT","SELECT")=
"HELP^CHECKING"
 ; accounts menu
 SET W("MAIN","M","ACCOUNTS","CHOICE",1)="&CHECKING"
 SET W("MAIN","M","ACCOUNTS","CHOICE",1,"EVENT","SELECT")=
"SEL^CHECKING"
 SET W("MAIN","M","ACCOUNTS","CHOICE",2)="&SAVINGS"
 SET W("MAIN","M","ACCOUNTS","CHOICE",2,"EVENT","SELECT")=
"SEL^CHECKING"
 ; file menu
 SET W("MAIN","M","FILE","CHOICE",1)="&NEW"
 SET W("MAIN","M","FILE","CHOICE",1,"EVENT","SELECT")=
 "NEW^CHECKING"
 SET W("MAIN","M","FILE","CHOICE",1,"SEPARATOR")=""
 SET W("MAIN","M","FILE","CHOICE",2)="&PRINT"
 SET W("MAIN","M","FILE","CHOICE",2,"EVENT","SELECT")=
"PRINT^CHECKING"
 SET W("MAIN","M","FILE","CHOICE",2,"ACTIVE")=0
 SET W("MAIN","M","FILE","CHOICE",2,"SEPARATOR")=""
 SET W("MAIN","M","FILE","CHOICE",3)="&SAVE"
 SET W("MAIN","M","FILE","CHOICE",3,"EVENT","SELECT")=
"FILE^CHECKING"
 SET W("MAIN","M","FILE","CHOICE",4)="SAVE &AS"
 SET W("MAIN","M","FILE","CHOICE",4,"ACTIVE")=0
 SET W("MAIN","M","FILE","CHOICE",4,"SEPARATOR")=""
 SET W("MAIN","M","FILE","CHOICE",5)="E&XIT"
 SET W("MAIN","M","FILE","CHOICE",5,"EVENT","SELECT")=
"CLOSE^CHECKING"
 ; gadgets
 ; check number
 SET W("MAIN","G","NUMBER","ACTIVE")=0
 SET W("MAIN","G","NUMBER","NEXTG")="SCROLL1
 SET W("MAIN","G","NUMBER","TYPE")="TEXT"
 SET W("MAIN","G","NUMBER","POS")="30,40"
 SET W("MAIN","G","NUMBER","SIZE")="50,25"
 SET W("MAIN","G","NUMBER","BCOLOR")="65535,65535,0"
 SET W("MAIN","G","NUMBER","TITLE")="CHECK"
 ; set the initial value with the last check number used + 1
 SET MWNUM=$P($G(^CHECK(0)),"^",2)+1
```

Example Seven

```
SET W("MAIN","G","NUMBER","VALUE")=MWNUM
; scroll for check number
SET W("MAIN","G","SCROLL1","TYPE")="SCROLL"
SET W("MAIN","G","SCROLL1","POS")="80,40"
SET W("MAIN","G","SCROLL1","SIZE")="20,25"
SET W("MAIN","G","SCROLL1","SCROLLDIR")="V"
SET W("MAIN","G","SCROLL1","SCROLLRANGE")="1,10000"
SET W("MAIN","G","SCROLL1","NEXTG")="AMOUNT"
SET W("MAIN","G","SCROLL1","VALUE")=MWNUM
SET W("MAIN","G","SCROLL1","EVENT","SELECT")="SCROLL^CHECKING"
; check amount
SET W("MAIN","G","AMOUNT","POS")="185,40"
SET W("MAIN","G","AMOUNT","SIZE")="50,25"
SET W("MAIN","G","AMOUNT","TYPE")="TEXT"
SET W("MAIN","G","AMOUNT","TITLE")="AMOUNT"
SET W("MAIN","G","AMOUNT","NEXTG")="SCROLL2"
; description of the transaction
SET W("MAIN","G","DESCRIPTION","POS")="185,120"
SET W("MAIN","G","DESCRIPTION","SIZE")="200,25"
SET W("MAIN","G","DESCRIPTION","TYPE")="TEXT"
SET W("MAIN","G","DESCRIPTION","TITLE")="DESCRIPTION"
SET W("MAIN","G","DESCRIPTION","NEXTG")="ACCOUNTS
; date - month
SET W("MAIN","G","MONTH","TYPE")="TEXT"
SET W("MAIN","G","MONTH","POS")="30,120"
SET W("MAIN","G","MONTH","SIZE")="30,25"
SET W("MAIN","G","MONTH","ACTIVE")=0
; date - day
SET W("MAIN","G","DAY","TYPE")="TEXT"
SET W("MAIN","G","DAY","POS")="60,120"
SET W("MAIN","G","DAY","SIZE")="30,25"
SET W("MAIN","G","DAY","ACTIVE")=0
; date - year
SET W("MAIN","G","YEAR","TYPE")="TEXT"
SET W("MAIN","G","YEAR","POS")="90,120"
SET W("MAIN","G","YEAR","SIZE")="30,25"
SET W("MAIN","G","YEAR","ACTIVE")=0
; values for MONTH, DAY and YEAR
SET MWDATE=$$MMDDYY($H)
SET MWMTH=$P(MWDATE,"/",1),MWDAY=$P(MWDATE,"/",2)
SET MWYR=$P(MWDATE,"/",3)
SET W("MAIN","G","MONTH","VALUE")=MWMTH
SET W("MAIN","G","DAY","VALUE")=MWDAY
SET W("MAIN","G","YEAR","VALUE")=MWYR
; scroll for date
SET W("MAIN","G","SCROLL2","TYPE")="SCROLL"
SET W("MAIN","G","SCROLL2","POS")="120,120"
SET W("MAIN","G","SCROLL2","SIZE")="20,25"
SET W("MAIN","G","SCROLL2","SCROLLDIR")="V"
SET W("MAIN","G","SCROLL2","SCROLLRANGE")="-30,30"
SET W("MAIN","G","SCROLL2","NEXTG")="DESCRIPTION"
SET W("MAIN","G","SCROLL2","EVENT","SELECT")="SCROLL^CHECKING"
; reconciled group
SET W("MAIN","G","RECONCILED","TYPE")="RADIO"
SET W("MAIN","G","RECONCILED","POS")="30,200"
SET W("MAIN","G","RECONCILED","SIZE")="120,110"
SET W("MAIN","G","RECONCILED","TITLE")="RECONCILED"
```

Example Seven

```
 SET W("MAIN","G","RECONCILED","CHOICE",1)="YES"
 SET W("MAIN","G","RECONCILED","CHOICE",2)="NO"
 SET W("MAIN","G","RECONCILED","NEXTG")="OK"
 ; list of accounts
 DO ACCOUNTS ; get choices in alphabetical order
 SET W("MAIN","G","ACCOUNTS","TYPE")="LIST"
 SET W("MAIN","G","ACCOUNTS","POS")="185,200"
 SET W("MAIN","G","ACCOUNTS","SIZE")="165,115"
 SET W("MAIN","G","ACCOUNTS","TITLE")="LIST OF ACCOUNTS"
 SET W("MAIN","G","ACCOUNTS","NEXTG")="RECONCILED"
 ; ok button
 SET W("MAIN","G","OK","TYPE")="BUTTON"
 SET W("MAIN","G","OK","POS")="380,200"
 SET W("MAIN","G","OK","SIZE")="80,40"
 SET W("MAIN","G","OK","TITLE")="OK"
 SET W("MAIN","G","OK","NEXTG")="CANCEL"
 SET W("MAIN","G","OK","EVENT","SELECT")="FILE^CHECKING"
 ; cancel button
 SET W("MAIN","G","CANCEL","TYPE")="BUTTON"
 SET W("MAIN","G","CANCEL","POS")="380,270"
 SET W("MAIN","G","CANCEL","SIZE")="80,40"
 SET W("MAIN","G","CANCEL","TITLE")="CANCEL"
 SET W("MAIN","G","CANCEL","EVENT","SELECT")="RESTORE^CHECKING"
 SET W("MAIN","EVENT","CLICK")="POPUP^CHECKING"
 SET W("MAIN","EVENT","CLICK","FILTERIN")="PB3"
TIMER ; set the inactivity timer - close if no activity
 SET MWTIME=300
 SET W("MAIN","T","TIMER","INTERVAL")=MWTIME
 SET W("MAIN","T","TIMER","EVENT","TIMER")="CHKACT^CHECKING"
 ; account notification label
 SET W("MAIN","G","NOTE","TYPE")="LABEL"
 SET W("MAIN","G","NOTE","POS")="300,40"
 SET W("MAIN","G","NOTE","SIZE")="120,20"
 SET W("MAIN","G","NOTE","TITLE")="CHECKING"
 SET W("MAIN","G","NOTE","FCOLOR")="65535,0,0"
 MERGE ^$W("MAIN")=W("MAIN") ; create window main
 ; date - label
 SET W("MAIN","G","DATE","TYPE")="LABEL"
 SET W("MAIN","G","DATE","POS")="30,100"
 ; get details from the abstraction layer
 SET MWFFACE=^$DI($PD,"FFACE")
 SET MWFSTYLE=^$DI($PD,"FSTYLE")
 SET MWFSIZE=^$DI($PD,"FSIZE")
 SET MWUNITS=^$DI($PD,"UNITS")
 SET X=$WTWIDTH("DATE",MWFFACE,MWFSIZE,MWFSTYLE,MWUNITS)
 SET Y=+$WFONT(MWFFACE,MWFSIZE,MWFSTYLE,MWUNITS)
 SET W("MAIN","G","DATE","SIZE")=X_","_Y ; area needed to display "DATE"
 SET W("MAIN","G","DATE","TITLE")="DATE"
 MERGE ^$W("MAIN","G","DATE")=W("MAIN","G","DATE") ; create gadget DATE
 SET ^$DI($PD,"FOCUS")="MAIN" ; set focus
 ESTART 9000 ; 9000 seconds is the time-out for event processing
 QUIT
 ;
MMDDYY(X) ; get month,day and year
 SET %DN=X DO 400^%DO
 QUIT %DN
```

Example Seven

for this article has each entry point from the event queue call a common tag to reference the ^$Event SSVN for details of the event. Example Six shows the tag called from the event node in Example Five, and the tag called to reference the event object.

After processing the event, control returns to the event loop to await another triggered event. The method for the detection and processing of events is referred to as, "Call Back Processing". This form of event processing can be imagined as a call to a routine after each READ to process input from the READ.

Managing the functions of windows and elements is a snap when using the MWAPI, since the operations of many objects are automatic. For example: resizing a window, word processing actions of text and document gadgets, and the scrolling of a list in a list box occur without the developer having to write code to support such actions.

The Interface shown in Illustration Two demonstrates a GUI created using the MWAPI in a Microsoft Windows 3.1 windowing environment. The interface shown in Illustration three has the characteristics of an interface seen when using Microsoft's Windows NT 4.0. The source code was not altered for the port. All the definitions for the interface are assigned to a local array prior to merging them into the ^Window SSVN. A global can be created to keep a persistent interface definition where all that is needed to create the GUI is Merge logic in the server application. For the experienced M programmer, changing from engineering applications that write to and read from character cell windows to the development of GUI applications requires minimal changes to already familiar syntax. The skill set developed while working with standard M applications can be transferred to MWAPI development. The MWAPI merely builds upon the specification for standard M to add the ability to create GUI clients and a connection to their server applications. Only a few SSVNs, special variables and string functions are added to the language syntax for the MWAPI. Developing with the MWAPI is a conceptual difference from character cell programming, not a syntactical one.

You can learn more about how to use the MWAPI to engineer real world applications with a full explanation for the code in Example Seven from *Reference MWAPI*. *Reference MWAPI* is available from the MTA. See the publications section of M computing, or visit the MTA web site at http://www.mtechnology.org and click on publications.

Reference MWAPI details how an M programmer can easily learn to develop robust window applications using the MWAPI. Included are numerous tables, illustrations, sample programs, images and discussions that demonstrate how high-quality graphical user interface applications can be easily developed. It describes how applications created through the host-independent development environment can be ported to any windows platform without changes in source code while maintaining the native look and feel, and the event processing methodology. It examines the issues of portability and compact code, some of the traits the MWAPI shares with the JAVA programming language, and which account for JAVA's popularity.

Reference MWAPI includes appendices designed for quick working reference, and it is organized with the needs of a developer in mind. The material is structured to serve as a learning tool and as a resource for applications development. The integration of the MWAPI is divided into seven chapters that focus on particular components of the system. The book is intended for the novice programmer as well as the more experienced M programmer. Sample programs walk the user through "how to" demonstrations of windows development. Exercises are included at the end of every chapter, and answers to the exercises are contained in Appendix B. The last chapter brings all the components together with a complete and detailed application. **M**

James Hay is a Senior Programmer/Analyst in Albuquerque, NM, and author of the book Reference MWAPI. *He can be reached by email at albqhay@gte.net.*

Additional reading available from the M Technology Association:

ANSI/MDC X11.6-1996 American National Standard for Information Systems - Programming Languages - M Windowing API, also published as ISO/IEC 15852:1999 Information Technology - Programming Languages - M Windowing API

Gardner, Guy, "Peeking at the New M Windowing API", *M Computing*, April 1993

Hay, James, *Reference MWAPI*, Boston, Digital Press, 1998

Trask, Gardner, "M Windowing API: Expansive Tool or Expensive Toy?", *M Computing*, February 1994.

Trask, Gardner, "The M Windowing API: The Tools", *M Computing*, April 1994.