FoxPro WorkGroup Extensions Help

Overview

Schedule+ Library

Message Function	Description
FMVersion	Returns the version of the FoxMAPI library.
SPReadMeet	Reads a meeting message.
SPSendMeet	Sends a meeting message.

Calendar Function	Description
SPBegin	Begins a calendar session and provides a session handle.
SPDelete	Deletes an item on a user's schedule.
SPEnd	Ends a calendar session.
SPFindNext	Returns the ID and type of the next (or first) item matching specified criteria.
SPFreeBusy	Returns merged free/busy information for a list of users.
SPReadAppt	Reads an appointment on a user's schedule.
SPReadTask	Reads an existing task on a user's schedule.
SPUserInfo	Returns information about a user.
SPSaveAppt	Creates a new appointment or modifies an existing one.
SPSaveTask	Creates a new task or modifies an existing one.

Cursor	Description
SPlusAppt	Specifies information associated with a calendar appointment.
SPlusAssoc	Specifies an item (such as an alarm) associated

	with an appointment or task.
SPlusAttd	Specifies a meeting attendee's identity and confirmation status.
SPlusMesg	Contains message information for meeting messages.
SPlusRest	Specifies an enumeration restriction passed to the SPFindNext function.
SPlusTask	Specifies information associated with a calendar task.
SPlusUser	Specifies information about a user.

MAPI Library

Function	Description
FMVersion	Returns the version of the FoxMAPI library.
MPAddress	Addresses a mail message.
MPCursor	Creates a MAPI Cursor.
MPDelete	Deletes a mail message.
MPDetails	Displays a recipient details dialog box.
MPFindNext	Returns the ID of the next (or first) mail message of a specified type.
MPLogoff	Ends a session with the messaging system.
MPLogon	Begins a session with the messaging system.
MPReadMail	Reads a mail message.
MPResolve	Displays a dialog box to resolve an ambiguous recipient name.
MPSaveMail	Saves a mail message.
MPSendDocs	Sends a standard mail message using a dialog box.
MPSendMail	Sends a mail message, allowing greater flexibility than MPSendDocs in message generation.

Cursor	Description
MapiFile	Contains file attachment information.
MapiMesg	Contains message information.
MapiRecip	Contains recipient information.

Overview

Scenarios

FoxMAPI

The simple messaging application program interface is a subset of the powerful function group known as MAPI, a set of messaging functions that help you create mail-enabled applications. With simple MAPI functions, you can easily add messaging to any Windows application. Some MAPI functions include a user interface (a dialog box), but you can also call MAPI functions without generating a user interface.

The FoxPro MAPI interface consists of a library (FOXMAPI.FLL). FOXMAPI.FLL is a FoxPro specific version of the Visual Basic MAPI library altered to accommodate FoxPro data structures.

FoxSPLUS

The Schedule+ Libraries allow you to develop applications that manipulate appointments and tasks on users' calendars, and exchange meeting messages between Schedule+ users. Portions of Schedule+ Libraries draw upon MAPI functions. As with MAPI, C language versions of the Schedule+ Libraries functions are available.

The Schedule+ Libraries function set is divided into two categories: meeting message functions and calendar functions.

Meeting message functions allow your application to send the following five kinds of messages:

- A meeting request requests or reschedules a meeting.
- A positive meeting response indicates that an attendee plans to attend a meeting.
- A negative meeting response indicates that an attendee won't attend a meeting.
- A tentative meeting response indicates that an attendee might attend a meeting.
- A meeting cancellation message rescinds meeting requests.

Calendar functions allow you to develop applications that manipulate appointments and tasks on users' schedules. With these applications, users can perform some or all of these operations:

- Create new appointments or tasks or modify existing ones.
- Delete appointments or tasks.
- List all appointments on a per-day basis, and list all tasks or all active tasks.

· Read information about a specific appointment or task.

Schedule+ Libraries also allow users of your application to determine if other users are free or busy at specific times and identify other users and their assistants.

In this version, Schedule+ Libraries do not fully handle the manipulation of recurring appointments or recurring tasks. However, you can use **SPFindNext** to list recurring appointments and recurring tasks, and you can use **SPDelete** to delete recurring appointments and recurring tasks.

Scenarios

To gain an understanding of what you can do with MAPI, consider these example scenarios:

<ins>Scenario 1 -- Sending a Bulk Mailing</ins>

<ins>Scenario 2 -- Sending Files in a Mail Message</ins>

<ins>Scenario 3 -- Creating FoxPro Applications that Read, Write, and Process Mail Messages</ins>

Sending a Bulk Mailing

You may want to send a mail message to a list of recipients. The actual content of the mail message could be a variety of things, such as a simple text message that is the same for all the recipients. Or it might be the contents of a Memo field that is merged with the name of each recipient to provide a custom mail message. You could even include documents, spreadsheets, bitmaps, and other files in your messages.

In the sample program, ***BULK.APP***, a mail message is sent to a list of recipients where the list of recipients is stored in a FoxPro table. The content of each message is a simple text message and a table of data. The table of data is different for each recipient based on the data for that recipient in the table.

Sending Files in a Mail Message

You may want to send a message that contains files that the recipients can then save to their own machines to work with. This is known as "attaching" files in your mail message.

For FoxPro developers, this feature might be useful when multiple developers are working on the same project and need to have the latest files for that project. Typically, each developer will have a local copy of the project and all its files. Each developer is responsible for working on a distinct set of files. When a developer makes changes, the new files need to be sent to the rest of the team.

The sample program, ***ATTACH.APP***, uses FoxPro tables to store the names of the developers on the project and the names of the files each developer is responsible for. A developer can run this sample code and then choose from a pick list of their files. FoxPro will attach these files and send them in a mail message to the other developers. The recipients can then "unattach" these files and rebuild their project.

Creating FoxPro Applications that Read, Write, and Process Mail Messages

You can write powerful FoxPro applications that send, receive, and process mail messages. These applications will use the Microsoft Mail backbone to deliver and store the mail messages but your FoxPro application would send and receive the messages in such a way that no other application (including the Microsoft Mail client) would be able to see those messages.

Differences Between FoxPro and C

The FoxMAPI interface library is very similar to the C MAPI interface. The function names have been changed slightly to conform with the Fox 10 character limit on FLL functions. Table 1 shows the mapping of the FoxMAPI function names to their C counterparts. The main variation from the C interface is the use of Fox cursors to simulate structured types in the FoxPro interface. The rest of the MAPI and SPLUS structures are mapped to FoxPro cursors as described in tables 2 through 10.

Table 1	FoxMAPI Function Names
Table 2	FoxMAPI MapiMesg Cursor Definition
Table 3	FoxMAPI MapiRecip Cursor Definition
Table 4	FoxMAPI MapiFile Cursor Definition
Table 5	FoxMAPI SPlusAppt Cursor Definition
Table 6	FoxMAPI SPlusAssoc Cursor Definition
Table 7	FoxMAPI SPlusAttd Cursor Definition

Cursors are passed to the FoxMAPI interface by alias name, i.e. by character string. The FoxMAPI functions will create an empty cursor if it does not already exist. Besides cursors, FoxMAPI deals with three other data types - numeric, character pointers, and array of character pointers. Numerics are passed as FoxPro numbers, character pointers are passed as FoxPro strings, and an array of character pointers are passed as an array of FoxPro strings.

Table 1 - FoxMAPI Function Names

C MAPI Name	FoxMAPI Name
MAPILogon	MPLogon
MAPILogoff	MPLogoff
MAPIFindNext	MPFindNext
MAPIReadMail	MPReadMail
MAPISaveMail	MPSaveMail
MAPIDeleteMail	MPDelete
MAPISendMail	MPSendMail
MAPISendDocument	MPSendDocs
MAPIAddress	MPAddress
MAPIResolveName	MPResolve
MAPIDetails	MPDetails
SPLUSBeginSession	SPBegin
SPLUSEndSession	SPEnd
SPLUSReadMeeting	SPReadMeet
SPLUSSendMeeting	SPSendMeet
SPLUSFindNext	SPFindNext
SPLUSDeleteItem	SPDelete
SPLUSReadAppt	SPReadAppt
SPLUSReadTask	SPReadTask
SPLUSReadUserInfo	SPUserInfo
SPLUSSaveAppt	SPSaveAppt

SPLUSSaveTask	SPSaveTask
SPLUSReadFreeBus	SPFreeBusy

Table 2 - FoxMAPI MapiMesg Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
Subject	Subject	C(254)
NoteText	NoteText	M
MessageType	MessagType	C(254)
DateReceived	DateRecved	C(16)
ConversationID	ConvertnID	C(254)
Flags	Flags	N(10,0)
RecipCount	RecipCount	N(10,0)
FileCount	FileCount	N(10,0)

Table 3 - FoxMAPI MapiRecip Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
RecipClass	RecipClass	N(10,0)
Name	Name	C(254)
Address	Address	C(254)
EIDSize	EIDSize	N(10,0)
EntryID	EntryID	M

Table 4 - FoxMAPI MapiFile Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
Flags	Flags	N(10,0)
Position	Position	N(10,0)
PathName	PathName	C(254)

FileName	FileName	C(254)
FileType	FileType	C(254)

Table 5 - FoxMAPI SPlusAppt Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
ItemType	ItemType	C(254)
Flags	Flags	N(10,0)
AssocCount	AssocCount	N(10,0)
OrganizerItemID	OrgItemID	C(254)
AttendeeCount	AttendCnt	N(10,0)
Text	Text	M
Body	Body	M
Recurrence	Recurrence	C(254)
DateStart	DateStart	C(16)
DateEnd	DateEnd	C(16)

Table 6 - FoxMAPI SPlusAssoc Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
ItemType	ItemType	C(254)
Flags	Flags	N(10,0)
ItemID	ItemID	C(254)
Position	Position	N(10,0)
Data	Data	C(254)

Table 7 - FoxMAPI SPlusAttd Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)

UserType	UserType	C(254)
Status	Status	N(10,0)

Table 8 - FoxMAPI SPlusMesg Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
Subject	Subject	C(254)
NoteText	NoteText	M
MessageType	MessagType	C(254)
DateReceived	DateRecved	C(16)
ConversationID	ConvertnID	C(254)
Flags	Flags	N(10,0)
RecipCount	RecipCount	N(10,0)
FileCount	FileCount	N(10,0)
SentForCount	SentForCnt	N(10,0)

Table 9 - FoxMAPI SPlusRest Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
ItemType	ItemType	C(254)
RestrictionType	RestType	C(254)
RestrictionData	RestData	C(254)

Table 10 - FoxMAPI SPlusTask Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
ItemType	ItemType	C(254)
Flags	Flags	N(10,0)
AssocCount	AssocCount	N(10,0)
OrganizerItemID	OrgItemID	C(254)

AttendeeCount	AttendCnt	N(10,0)
Text	Text	M
Body	Body	M
Recurrence	Recurrence	C(254)
DateDue	DateDue	C(16)
DurationActive	DuratActiv	C(6)
ProjectName	ProjctName	C(254)
Priority	Priority	C(1)

Table 11 - FoxMAPI SPlusUser Cursor Definition

C Name	FoxMAPI Name	Type
Reserved	Reserved	N(10,0)
UserType	UserType	C(254)
Flags	Flags	N(10,0)
StartTime	StartTime	C(254)
EndTime	EndTime	C(254)
TimeZone	TimeZone	C(254)
Data	Data	C(254)

Distributing MAPI/SPLUS Applications

The only difficulty in distributing applications that use **MAPI.DLL**, **SPLUS.DLL**, or **MEFLIB.DLL** is getting the setup program for your application to install these libraries in each users' \windows\system directory.

The WorkGroup Extensions for FoxPro come with a file called **REQUIRED.DBF**. This is a new version of a file that the Setup Wizard uses to install a series of required DLLs into each users' system directory.

To install the library files to each user's \windows\system directory, follow these steps:

1 Copy the three files **MAPI.DL_**, **SPLUS.DL_**, **MEFLIB.DL_** (found in the libs directory of the WorkGroup Extension files) into the directory where you installed the FoxPro Distribution Kit for Windows (the default is c:\foxprow\dksetup).

2 Copy the file **REQUIRED.DBF** (found in the libs directory of the WorkGroup Extension files) into the directory where the Setup Wizard was installed (the default is c:\foxprow\wizard). Note that you will be copying over the older REQUIRED.DBF.

3 Launch FoxPro for Windows.

4 Open the project called **setup.pjx** found in the directory where the Setup Wizard was installed (the default is c:\foxprow\wizard).

5 Rebuild the app called **setup.app** that, by default, is placed in the c:\foxprow directory.

6 Close **setup.pjx** and quit FoxPro for Windows.

Now everytime you build a setup program using the Setup Wizard, the three library files will be copied to your users' windows\system directory. Note that if a user has a later version of any of these files, the older version will not be installed, ensuring that the most up to date version of these DLLs is always on the users' machines.

It is a good idea to create a backup copy of the original REQUIRED.DBF before copying over it with the version that comes with WorkGroup Extensions. By creating a backup copy, you can swap in whichever REQUIRED.DBF you want and rebuild the Setup Wizard depending on whether or not you want to install these DLLs on your user's machines. You can also get the original REQUIRED.DBF by reinstalling the Distribution Kit.

Finally, when you receive a newer version of any of these three DLLs, you will want to edit that DLL's record in REQUIRED.DBF, updating the FILSIZE, EXPNDSIZE, FDATE, and VERSION fields. When you are finished, rebuild the Setup Wizard.

FMVersion

Returns the version of the current FoxMAPI library.

Parameters

None.

Returns

Version of the library as a string.

MapiFile

{bmc seealso.bmp}

```
Cursor MapiFile
        Reserved as Numeric
        Flags as Numeric
        Position as Numeric
        PathName as String
        FileName as String
        FileType as String
End Cursor
```

Description

The **MapiFile** type contains information about a file attachment. Simple MAPI supports the following kinds of attachments:

{bmc b.bmp} Data files
{bmc b.bmp} Embedded OLE objects
{bmc b.bmp} Static OLE objects

The *Flags* field determines the kind of attachment. Object linking and embedding (OLE) object files are file representations of OLE object streams. You can re-create an OLE object from the file by calling the **OleLoadFromStream** function with an OLESTREAM object that reads the file contents. (See the Microsoft Windows SDK documentation for details.) If an OLE file attachment is included in an outbound message, the OLE object stream should be written directly to the file used as the attachment.

Field	**Description**	**Type**
Reserved	Reserved for future use. This field must be 0.	N(10,0)
Flags	A bitmask of flags. Unused flags are reserved and must be 0. The following flags are defined in MAPILIB.PRG: #define MP_OLE 1 #define MP_OLE_STATIC 2 Set MP_OLE if the attachment is an OLE object. Also set MP_OLE_STATIC if the attachment is a static OLE object rather than an embedded OLE object	N(10,0)
Position	An integer describing where the	N(10,0)

	attachment should be placed in the message body. Attachments replace the character found at a certain position in the message body; in other words, attachments replace the **MapiMesg** type field *NoteText*[*Position*]. Applications cannot place two attachments in the same location within a message, and attachments cannot be placed beyond the end of the message body.	
PathName	The full path name of the attached file. The file should be closed before this call is made.	C(254)
FileName	The filename seen by the recipient. This name can differ from the filename in *PathName* if temporary files are being used. If *FileName* is empty, the filename from *PathName* is used. If the attachment is an OLE object, *FileName* contains the class name of the object; for example, “Microsoft Excel Worksheet."	C(254)
FileType	A reserved descriptor that describes to the recipient the type of the attached file. An empty string indicates an unknown or operating system-determined file type. With this release, you must use "" for this parameter.	C(254)

Comments

MPReadMail does not return an attachment with *Position* equal to -1 unless you set the MP_BODY_AS_FILE flag.

With Microsoft Mail version 3.0b, if you send an attachment with *Position* equal to -1, the attachment is placed at the beginning of the message, with a space character on either side of the attachment.

See Also

MapiMesg

MPReadMail

MapiRecip

MapiMesg

{bmc seealso.bmp}

```
Cursor MapiMesg
    Reserved as Numeric
    Subject as String
    NoteText as String
    MessageType as String
    DateReceived as String
    ConversationID as String
    Flags as Numeric
    Originator as Numeric
    RecipCount as Numeric
    FileCount as Numeric
End Cursor
```

Description

The **MapiMesg** type contains information about a message.

Field	Description	Type
Reserved	Reserved for future use. This field must be 0.	N(10,0)
Subject	The subject text, limited to 256 characters or less. (Messages saved using **MPSaveMail** are not limited to 256 characters.) An empty string indicates no subject text.	C(254)
NoteText	A string containing text in the message. An empty string indicates no text. For inbound messages, each paragraph is terminated with a carriage return-line feed pair (0x0d0a). For outbound messages, paragraphs can be delimited with a carriage return, a line feed, or a carriage return-line feed pair (0x0d, 0x0a, or 0x0d0a).	M
MessageType	A message type string used by applications other than interpersonal electronic mail. An empty string indicates an interpersonal message (IPM) type.	C(254)

DateReceived	A string indicating the date a message is received. The format is YYYY/MM/DD HH:MM; hours are measured on a 24-hour clock.	C(16)
ConversationID	A string indicating the conversation thread ID to which this message belongs.	C(254)
Flags	A bitmask of flags. Unused flags are reserved. Unused flags must be 0 for outbound messages and are ignored for inbound messages. The following flags are defined in MAPILIB.PRG: #define MP_UNREAD 1 #define MP_RECEIPT_REQUESTED 2 #define MP_SENT 4	N(10,0)
Originator	A **MapiFile** type that describes the message originator.	N(10,0)
RecipCount	A count of the recipient descriptor types pointed to by *Recips.* A value of 0 indicates that no recipients are included.	N(10,0)
FileCount	A count of the file attachment descriptor types pointed to by *Files.* A value of 0 indicates that no file attachments are included.	N(10,0)

Comments

Microsoft Mail supports message types (the *MessageType* parameter) that begin with IPM or IPC. Messages of type IPM (interpersonal mail) are visible in the Inbox when delivered. Messages of type IPC (interprocess communication) are not displayed in the Inbox when delivered. For more information about message types, see the programmer's reference that accompanies this product.

See Also

MapiFile

MapiRecip

MapiRecip

{bmc seealso.bmp}

```
Cursor MapiRecip
    Reserved as Numeric
    RecipClass as Numeric
    Name as String
    Address as String
    EIDSize as Numeric
    EntryID as String
End Cursor
```

Description

The **MapiRecip** type contains information about a message originator or recipient.

Field	Description	Type
Reserved	Reserved for future use. This field must be 0.	N(10,0)
RecipClass	Classifies the recipient of the message. (Messages can be sorted by recipient class.) This field can also contain information about the originator of an inbound message.	N(10,0)
Name	The name of the recipient that is displayed by the messaging system.	C(254)
Address	Provider-specific message delivery data. This can be used by the message system to identify recipients who are not in an address list (“one-off” addresses).	C(254)
EIDSize	The size (in bytes) of the opaque binary data in EntryID.	N(10,0)
EntryID	A binary string used by the messaging system to efficiently specify the recipient. Unlike the Address field, this data is opaque and is not printable. The messaging system returns valid EntryIDs for recipients or originators included in the address list.	M

See Also

MapiFile

MapiMesg

MPAddress

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPAddress(
    Session as Numeric,
    UIParam as Numeric,
    Caption as String,
    EditFields as Numeric,
    Label as String,
    RecipCount as Reference to a Numeric,
    Recipients() as MapiRecip,
    Flags as Numeric,
    Reserved as Numeric) as Numeric
```

Description

With this function, users can create or modify a set of address list entries using a standard address list dialog box. The dialog box cannot be suppressed, but function parameters allow the caller to set characteristics of the dialog box.

The call is made with an initial, and possibly empty, set of recipients. The address list dialog box shows the contents of the recipient set; users can choose new entries to add to the set. The final set of recipients is returned to the caller in *RecipCount* and *Recipients*, destroying their initial values.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. The session handle is returned by **MPLogon** and invalidated by **MPLogoff**. If the value is 0, the messaging system initiates a session from a system default session (if one exists) or presents a sign-in dialog box. In all cases, the messaging system returns to its state before the call.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.

Caption	The caption of the address list dialog box. If this parameter is an empty string, the default value "Address Book" is used.
EditFields	The number of edit controls that should be present in the address list. The values 0 to 4 are valid. If *EditFields* is 0, only address-list browsing is allowed. *EditFields* values of 1 to 3 control the number of edit controls present. Entries selected for the different controls are differentiated by the *RecipClass* field in the returned recipient type. If *EditFields* is 4, each recipient class supported by the underlying messaging system has an edit control. If the number of recipient classes in *Recips* is greater than the value of *EditFields,* the number of classes in *Recips* is used instead of *EditFields.* If *EditFields* is 1 and more than one kind of entry exists in *Recips*, *Labels* is ignored.
Label	A string used as an edit control label in the address list dialog box. This argument is ignored and should be an empty string except when *EditFields* is 1. If you want a default control label "To:", *Label* should be an empty string.
RecipCount	The number of entries in *Recipients.* If *RecipCount* is 0, *Recipients* is ignored.
Recipients( )	The initial array of recipient entries to be used to populate edit controls in the address list dialog box. Recipient entries need not be grouped by recipient class. If the value of the greatest recipient class present is greater than the *EditFields* parameter, *EditFields* and *Label* are ignored. This array is redimensioned as necessary to accommodate the entries made by the user in the address list dialog box.
Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG:

```
#define MP_LOGON_UI   1
' Display logon UI
#define MP_NEW_SESSION   2
' Don't get default if available
```

Set MP_LOGON_UI if the function should display a sign-in dialog box (if required). When this flag is not set, the function does not display a dialog box and

	returns an error if the user is not signed in. If the session passed in *Session* is not 0, this flag is ignored.
	Set MP_NEW_SESSION if you want to establish a session other than the current one. For instance, if a mail client is already running, another MAPI electronic-mail client can piggyback on the session created by the mail client application. Do not set this flag if you want the default session (if it still exists). If the session passed in *Session* is not 0, this flag is ignored.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	**Meaning**
MP_E_FAILURE	One or more unspecified errors occurred while addressing the mail. No list of entries was returned.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No list of entries was returned.
MP_E_INV_EDITFIELDS	The value of *EditFields* was outside the range of 0 to 4. No list of entries was returned.
MP_E_INV_MESSAGE	An invalid message ID was used for the *MessageID* parameter. No list of entries was returned.
MP_E_INV_RECIPS	One or more of the recipients in the address list were not valid. No list of entries was returned.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No list of entries was returned.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No list of entries was returned.
MP_E_NOT_SUPPORTED	The operation was not supported by the underlying messaging system.
MP_USER_ABORT	The user canceled the process. No list of entries was returned.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPLogoff

MPLogon

MPDelete

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPDelete(
    Session as Numeric,
    UIParam as Numeric,
    MessageID as String,
    Flags as Numeric,
    Reserved as Numeric) as Numeric
```

Description

This function deletes a message from the message store. Before calling **MPDelete,** use **MPFindNext** to verify that the message to be deleted is the one you want deleted.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. The session handle is returned by **MPLogon** and invalidated by **MPLogoff**. If the value is 0, the messaging system establishes a session from a system default session (if one exists) or presents a sign-in dialog box. In all cases, the messaging system returns to its state before the call.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
MessageID	The messaging system's string identifier for the message being deleted. The string identifier is returned by **MPFindNext** or **MPSaveMail**. Applications should assume that this identifier is invalid after **MPDelete** returns successfully.
Flags	A bitmask of flags. All flags are reserved and should be set to 0.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	**Meaning**
MP_E_FAILURE	One or more unspecified errors occurred while deleting the mail. No mail was deleted.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No mail was deleted.
MP_E_INV_MESSAGE	An invalid message ID was used for the *MessageID* parameter. No mail was deleted.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No mail was deleted.
MP_USER_ABORT	The user canceled the process. No mail was deleted.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPFindNext

MPLogoff

MPLogon

MPSaveMail

MPDetails

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPDetails(
    Session as Numeric,
    UIParam as Numeric,
    Recipient as MapiRecip,
    Flags as Numeric,
    Reserved as Numeric) as Numeric
```

Description

This function presents a dialog box that provides the details of a given address list entry. The dialog box cannot be suppressed. At the option of the caller, the entry can be either modifiable or fixed. The call works only for names that have resolved either as the recipients of read mail, resolved entries returned by **MPAddress**, or entries returned by **MPResolve**.

The directory the entry belongs to determines the amount of information presented in the details dialog box. It contains at least the display name and address of the recipient.

An entry is resolved if the *EIDSize* field of the **MapiRecip** type is nonzero.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. Session handles are returned by **MPLogon** and invalidated by **MPLogoff**. If the value is 0, the messaging system sets up a session from a system default session (if one exists) or presents a sign-in dialog box. In all cases, the messaging system returns to its state before the call.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
Recipient	A recipient descriptor containing the entry whose details are to be displayed. All fields of the **MapiRecip** type except *EIDSize* and *EntryID* are ignored. If the field *EIDSize* is zero, MP_E_AMB_RECIP is returned.
Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG:

```
#define MP_LOGON_UI   1
' Display logon UI
#define MP_NEW_SESSION   2
' Don't get default if available
#define MP_AB_NOMODIFY   1024
' Don't modify PAB entries
```

Set MP_LOGON_UI if the function should display a sign-in dialog box (if required). When this flag is not set, the function does not display a dialog box and returns an error if the user is not signed in.

Set MP_NEW_SESSION if you want to establish a session other than the current one. For instance, if a mail client is already running, another MAPI electronic-mail client can piggyback on the session created by the mail client application. Do not set this flag if you want the default session (if it still exists). If the session passed in *Session* is not 0, this flag is

	ignored. Set MP_AB_NOMODIFY if the details of the entry should not be modifiable even if the entry belongs to the personal address book.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	**Meaning**
MP_E_AMB_RECIP	The *EIDSize* field of the *Recipient* parameter was zero. No dialog box was displayed.
MP_E_BAD_RECIPTYPE	One or more recipients were unknown. No dialog box was displayed.
MP_E_FAILURE	One or more unspecified errors occurred while matching the message type. The call failed before message type matching could take place.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No dialog box was displayed.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No dialog box was displayed.
MP_E_NOT_SUPPORTED	The operation was not supported by the underlying messaging system.
MP_USER_ABORT	The user canceled the process. No dialog box was displayed.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPAddress

MPLogoff

MPLogon

MapiRecip

MPResolve

MPFindNext

<u>{bmc retrnval.bmp}{bmc seealso.bmp}</u>

```
MPFindNext(
    Session as Numeric,
    UIParam as Numeric,
    MessageType as String,
    SeedMessageID as Numeric,
    Flags as Numeric,
    Reserved as Numeric,
    MessageID as Reference to a String) as Numeric
```

Description

This function allows an application to enumerate messages of a given type. It returns message identifiers that can be used in subsequent MAPI function calls to retrieve and delete messages. **MPFindNext** is designed to process incoming mail, not manage received mail. **MPFindNext** looks for messages in the folder in which new messages of the specified type are delivered. **MPFindNext** calls can be made only in the context of a valid MAPI session established with **MPLogon**.

When provided with an empty *SeedMessageID*, **MPFindNext** returns the ID of the first message specified with *MessageType.* When provided a non-empty *SeedMessageID*, **MPFindNext** returns the next matching message of the type specified with *MessageType*. Repeated calls to **MPFindNext** ultimately result in a return of MP_E_NO_MESSAGES, which means that the enumeration of the matching message types is complete.

Message identifiers are not guaranteed to remain valid, because other applications can move or delete messages. Applications must be able to handle failed calls to **MPFindNext**, **MPDelete**, and **MPReadMail** for invalid message IDs. The ordering of messages is system-specific. Message ID strings must be dynamic strings.

Message type matching is done against message class strings. All message types whose names match (up to the length specified in the *MessageType* parameter) are returned. If the *MessageType* parameter begins with the letters IPM, matching is performed in the Inbox. Otherwise, matching is performed in the hidden application mail folder. If the message type is an empty string, all messages in the Inbox are included in the list.

Parameter	Description

Session	An opaque session handle whose value represents a session with the messaging system. Session handles are returned by **MPLogon** and invalidated by **MPLogoff**. If the value is 0, **MPFindNext** returns MP_E_INV_SESSION. In all cases, the messaging system returns to its state before the call.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
MessageType	A pointer to a string that is the message type. To specify an interpersonal mail message use an empty string, "".
SeedMessageID	A string that is the message identifier seed for the request. If the identifier is an empty string, the first message matching the type specified in the *MessageType* parameter is returned. Message IDs are system-specific and opaque. Message IDs may be invalidated at any time if another application moves or deletes a message.
Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG:

```
#define MP_UNREAD_ONLY   32
' Only unread messages
#define MP_NEW_SESSION   2
' Don't get default if available
#define MP_GUARANTEE_FIFO   256
' Guarantee FIFO MPFindNext
```

Set MP_UNREAD_ONLY if the function should enumerate only unread messages. When this flag is not set, all messages of the given type are returned.

Set MP_NEW_SESSION if you want to establish a session other than the current one. For instance, if a mail client is already running, another MAPI electronic-mail client can piggyback on the session created by the mail client application. Do not set this flag if you want the default session (if it still exists). If the session passed in *Session* is not 0, this flag is ignored.

Set MP_GUARANTEE_FIFO if you want the message IDs returned in the order the messages were received. **MPFindNext** calls may take longer if this flag is set.

Reserved	Reserved for future use. This parameter must be 0.
MessageID	A variable-length string that is the message identifier. Message IDs are system-specific, nonprintable, and opaque. Message ID strings must be dynamic strings. Message IDs may be invalidated at any time if another application deletes or moves a message.

Return Value	**Meaning**
MP_E_FAILURE	One or more unspecified errors occurred while matching the message type. The call failed before message type matching could take place.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No mail was found.
MP_E_INV_MESSAGE	An invalid message ID was used for the *SeedMessageID* parameter. No mail was found.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No mail was found.
MP_E_NO_MESSAGES	The **MPFindNext** function could not find a matching message.
MP_USER_ABORT	The user canceled the process. No mail was found.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPDelete

MPLogoff

MPLogon

MPReadMail

MPLogoff

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPLogoff(
    Session as Numeric,
    UIParam as Numeric,
    Flags as Numeric,
    Reserved as Numeric) as Numeric
```

Description

This function ends a session with the messaging system.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. Session handles are returned by **MPLogon** and invalidated by **MPLogoff**.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
Flags	Reserved for future use. This parameter must be 0.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	Meaning
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. The session was not terminated.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. The session was not terminated.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPLogon

MPLogon

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPLogon(
    UIParam ByVal as Numeric,
    User as String,
    Password as String,
    Flags as Numeric,
    Reserved as Numeric,
    Session as Reference to a Numeric) as Numeric
```

Description

The **MPLogon** function begins a session with the messaging system. You can sign in to the messaging system in two ways using simple MAPI mail calls:

- Implicitly sign in.

  Any MAPI function call made outside of an established MAPI session generates a sign-in dialog box, which can be suppressed by the calling application. In this case, when the call returns, the session is terminated and the messaging system returns to its state before the call was made. For example, a user signed off from the messaging system before the call would also be signed off after the call completed.

- Explicitly sign in using the **MPLogon** function (and sign off using **MPLogoff**).

  If you want to maintain a session over a number of simple MAPI calls, you can use the **MPLogon** function to provide a session handle to the messaging system. This session handle can be used in subsequent MAPI calls to explicitly provide user credentials to the messaging system. A flag is available to display a sign-in dialog box if the credentials presented fail to validate the session. You can pass an empty password, although it may not validate the mail session.

Parameter	Description
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
User	A client account-name string, limited to 256 characters or less. An empty string indicates that a sign-in dialog box with an empty name field should be generated (if the appropriate flag is set).

Password	A credential string, limited to 256 characters or less. An empty string indicates that a sign-in dialog box with an empty password field should be generated (if the appropriate flag is set) or that the messaging system does not expect password credentials.
Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG:

```
#define MP_LOGON_UI   1
' Display logon dialog box
#define MP_NEW_SESSION   2
' Get default if available
#define MP_FORCE_DOWNLOAD   4096
' Force message download from server
```

Set MP_LOGON_UI if the function should display a dialog box to prompt for name and password (if required). When this flag is not set, the **MPLogon** function does not display a sign-in dialog box and returns an error if the user is not signed in.

Set MP_NEW_SESSION if you want to establish a session other than the current one. For instance, if a mail client is already running, another MAPI electronic-mail client can piggyback on the session created by the mail client application. Do not set this flag if you want the default session (if it still exists).

Set MP_FORCE_DOWNLOAD to force a download of all new messages from the mail server to a user's Inbox during the sign-in process. Use this flag so an application can deal with the user's complete set of messages when it signs in. When this flag is set, a progress indicator is displayed, and is automatically removed when the process is complete. Use of this flag may increase processing time.

Reserved	Reserved for future use. This parameter must be 0.
Session	An opaque session handle whose value is set by the messaging system when the **MPLogon** call is successful. The session handle can then be used in subsequent MAPI calls.

Return Value	**Meaning**
MP_E_FAILURE	One or more unspecified errors occurred during sign-in. No session handle was returned.

MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No session handle was returned.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No session handle was returned.
MP_E_TM_SESSIONS	The user had too many sessions open at once. No session handle was returned.
MP_USER_ABORT	The user canceled the process. No session handle was returned.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPLogoff

MPReadMail

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPReadMail(
    Session as Numeric,
    UIParam as Numeric,
    MessageID as String,
    Flags as Numeric,
    Reserved as Numeric,
    Message as MapiMesg,
    Originator as MapiRecip,
    Recips() as MapiRecip,
    Files() as MapiFile) as Numeric
```

Description

This function reads a mail message. Before calling **MPReadMail,** use **MPFindNext** to verify that the message to be read is the one desired.

The call returns one message, breaking the message content into the same parameters and types used in the **MPSendMail** function. **MPReadMail** fills a block of memory with the **MapiMesg** type containing message elements. File attachments are saved to temporary files, and the names are returned to the caller in the message type. Recipients, attachments, and contents are copied from the message before the function returns to the caller, so later changes to the files do not affect the contents of the message.

A flag is provided to specify that only envelope information is to be returned from the call. Another flag (in the **MapiMesg** type) specifies whether the message is marked as sent or unsent.

All strings are null-terminated and must be specified in the current character set or code page of the calling program's operating system process. In Microsoft Windows, the character set is ANSI.

The originator, recipients, and file attachments are written into the appropriate parameters of the FoxPro call. *Recips* and *Files* should be dynamically allocated arrays of their respective types.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. If the value is 0, **MPReadMail** returns MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
MessageID	A variable-length string that is the message identifier of the message to be read. Message IDs are system-specific, nonprintable, and opaque. Message IDs can be obtained from the **MPFindNext** and **MPSaveMail** functions.
Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG: `#define MP_ENVELOPE_ONLY   64` `' Only header information` `#define MP_SUPPRESS_ATTACH   2048` `' Header and body, no files` `#define MP_BODY_AS_FILE   512` `'Save body as first attachment` `#define MP_PEEK   128` `' Don't mark message as read` Set MP_ENVELOPE_ONLY if you don't want the function to copy attachments to temporary files or return the note text. All other message information (except for temporary filenames) is returned. Setting this flag usually reduces the processing time required for the function. Set MP_SUPPRESS_ATTACH if you don't want **MPReadMail** to copy attachments but just to return

	note text. If MP_ONLY_ENVELOPE is set, this flag is ignored. The flag should reduce the time required by the **MPReadMail** function.
	Set MP_BODY_AS_FILE if you want the function to write the message body to a temporary file and add it to the attachment list as the first attachment, instead of returning a pointer to the message body (the default behavior). The *Position* parameter of a body attachment is -1.
	Set MP_PEEK if you don't want **MPReadMail** to mark the message as read. Any unsuccessful return leaves the message unread.
Reserved	Reserved for future use. This parameter must be 0.
Message	A type set by **MPReadMail** to a message containing the message contents.
Originator	The originator of the message.
Recips ( )	An array of recipients. This array is redimensioned as necessary to accommodate the number of recipients chosen by the user.
Files ( )	An array of attachment files written when the message is read. When **MPReadMail** is called, all message attachments are written to temporary files. It is the caller's responsibility to delete these files when they are no longer needed. When MP_ENVELOPE_ONLY or MP_SUPPRESS_ATTACH is set, no temporary files are written and no temporary names are filled into the file attachment descriptors. This array is re-dimensioned as necessary to accommodate the number of files attached by the user.

Return Value	**Meaning**
MP_E_ATT_	
WRITE_FAILURE	An attachment could not be written to a temporary file. Check directory permissions.
MP_E_DISK_FULL	The disk was full.
MP_E_FAILURE	One or more unspecified errors occurred while reading the mail. No mail was read.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to

	proceed. No mail was read.
MP_E_INV_MESSAGE	The message ID was invalid. The message ID may have been deleted or changed by another process.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No mail was read.
MP_E_NOT_SUPPORTED	The operation was not supported by the underlying messaging system.
MP_E_TM_FILES	Too many file attachments were contained in the message. No mail was read.
MP_E_TM_RECIPIENTS	There were too many recipients of the message. No mail was read.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPFindNext

MPLogon

MapiMesg

MPSaveMail

MPSendMail

MPResolve

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPResolve(
    Session as Numeric,
    UIParam as Numeric,
    UserName as String,
    Flags as Numeric,
    Reserved as Numeric,
    Recipient as MapiRecip) as Numeric
```

Description

This function resolves a mail recipient's name (as entered by a user) to an unambiguous address list entry, optionally prompting the user to choose between ambiguous entries if necessary. A recipient descriptor is allocated and returned containing fully resolved information about the entry.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. The session handle is returned by **MPLogon** and invalidated by **MPLogoff**. If the value is 0, the messaging system initiates a session from a system default session (if one exists) or presents a sign-in dialog box. In all cases, the messaging system returns to its state before the call.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
UserName	A string containing the name to be resolved.
Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG:

```
#define MP_LOGON_UI   1
' Display logon UI
#define MP_NEW_SESSION   2
' Don't get default if available
#define MP_DIALOG   8
' Prompt to resolve ambig. names
#define MP_AB_NOMODIFY   1024
' User can't modify PAB entries
```

Set MP_LOGON_UI if the function should display a sign-in dialog box (if required). When this flag is not set, the function does not display a dialog box and returns an error if the user is not signed in.

Set MP_NEW_SESSION if you want to establish a session other than the current one. For instance, if a mail client is already running, another MAPI electronic-mail client can piggyback on the session created by the mail client application. Do not set this flag if you want the default session (if it still exists). If the session passed in *Session* is not 0, this flag is ignored.

Set MP_DIALOG if **MPResolve** should attempt to resolve names by displaying a name resolution dialog box to the user. If this flag is not set, resolutions which do not result in a single name will return MP_E_AMB_RECIP.

	Set MP_AB_NOMODIFY if the details of the entry should not be modifiable even if the entry belongs to the personal address book. (Part of the resolution dialog box could involve displaying details about the various entries that match the *UserName* parameter. Set this flag if these details should not be modifiable.) This flag is ignored if MP_DIALOG is not set.
Reserved	Reserved for future use. This parameter must be 0.
Recipient	A recipient type set by **MPResolve** if the resolution results in a single match. The type contains the recipient information of the resolved name. The descriptor can then be used in calls to **MPSendMail**, **MPSaveMail**, and **MPAddress**.

Return Value	**Meaning**
MP_E_AMB_RECIP	One or more recipients was specified ambiguously, or the *EIDSize* parameter was 0. The name was not resolved.
MP_E_FAILURE	One or more unspecified errors occurred while addressing the mail. The name was not resolved.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. The name was not resolved.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. The name was not resolved.
MP_E_NO_RECIP	No name was specified in the *UserName* parameter.
MP_E_NOT_SUPPORTED	The operation was not supported by the underlying messaging system.
MP_USER_ABORT	The user canceled the process. The name was not resolved.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPAddress

MPLogoff

MPLogon

MPSaveMail

MPSendMail

MPSaveMail

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPSaveMail(
    Session as Numeric,
    UIParam as Numeric,
    Message as MapiMesg,
    Recips as MapiRecip,
    Files as MapiFile,
    Flags as Numeric,
    Reserved ByVal as Numeric,
    MessageID as Reference to a String) as Numeric
```

Description

This function saves a message, optionally replacing an existing message. Before calling **MPSaveMail,** use **MPFindNext** to verify that the message to be saved is the correct one. *MessageID* must be a variable-length string. The elements of the message identified by the *MessageID* parameter are replaced by the elements in the *Message* parameter. If the *MessageID* parameter is empty, a new message is created. The new message ID is returned in the *MessageID* parameter on completion of the call. All replaced messages are saved in their appropriate folders. New messages are saved in the folder appropriate for incoming messages of that class.

The FoxPro **MPSaveMail** function takes the recipients and file attachments from the *Recips* and *Files* parameters, which should each be the first element of dynamically allocated arrays of their respective types. These arrays are not re-dimensioned and are not affected by assignment side-effects.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. Session handles are returned by **MPLogon** and invalidated by **MPLogoff**. If the value is 0 and *MessageID* is an empty string, the messaging system establishes a session from a system default session (if one exists) or presents a sign-in dialog box. Otherwise, calls with *Session* equal to 0 return MP_E_INV_SESSION.

UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
Message	A pointer to the **MapiMesg** type containing the contents of the message to be saved. The MP_SENT flag is ignored.
Recips	The first element in an array of recipients. When *RecipCount* is 0, this parameter is ignored. The recipient string can include either the recipient's name or the recipient's name-address pair. If just a name is specified, the name is resolved to an address using implementation-defined address book search rules. If an address is also specified, a search for the name is not performed. The address is in an implementation-defined format and is assumed to have been obtained from the implementation some other way. When the address is specified, the name is used for display to the user and the address is used for delivery. When *EntryID* is used, no search is performed and the display-name and address are ignored. (A name and address are associated with the *EntryID* within the messaging system.) *EntryID*s are returned by the **MPReadMail** function.
Files	The first element in an array of attachment files written when the message is read. The number of attachments per message may be limited in some systems. If the limit is exceeded, the error MP_E_TM_FILES is returned. When *FileCount* is 0, this parameter is ignored. Attachment files are read and attached to the message before the call returns. Do not attempt to display attachments outside the range of the message body.

Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG:
	`#define MP_LOGON_UI 1`
	`* Display logon UI`
	`#define MP_NEW_SESSION 2`
	`* Don't get default if available`
	Set MP_LOGON_UI if the function should display a sign-in dialog box (if required). When you do not set

	this flag, the function does not display a dialog box and returns an error if the user is not signed in.
	Set MP_NEW_SESSION if you want to establish a session other than the current one if supported by the messaging system. For instance, if a mail client is already running, another MAPI electronic-mail client can piggyback on the session created by the mail client application. Do not set this flag if you want the default session (if it still exists).
Reserved	Reserved for future use. This parameter must be 0.
MessageID	The variable-length string identifier for this message. It is returned by the **MPFindNext** function or a previous call to **MPSaveMail**. If a new message is to be created, this parameter should point to an empty string. Message ID strings must be dynamic strings.

Return Value	**Meaning**
MP_E_DISK_FULL	The disk was full.
MP_E_FAILURE	One or more unspecified errors occurred while saving the mail. No mail was saved.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No mail was saved.
MP_E_INV_MESSAGE	An invalid message ID was used for the *MessageID* parameter. No mail was saved.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No mail was saved.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No mail was saved.
MP_E_NOT_SUPPORTED	The operation was not supported by the underlying messaging system.
MP_USER_ABORT	The user canceled the process. No mail was saved.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPFindNext

MPLogoff

MPLogon

MPSendDocs

{bmc retrnval.bmp}

```
MPSendDocs(
    UIParam as Numeric,
    DelimChar as String,
    FilePaths as String,
    FileNames as String,
    Reserved as Numeric) as Numeric
```

Description

The **MPSendDocs** function sends a standard mail message. Calling the function displays a Send Note dialog box, which prompts the user to send a mail message with data file attachments. Attachments can include the active document or all the currently open documents in the Windows-based application that called **MPSendDocs**. The function is used primarily for calls from a macro or scripting language, often found in applications such as spreadsheet or word-processing programs.

The user's default sign-in identification is used when sending the mail. If there is no default identification when this function is called, a standard sign-in dialog box appears. After the user provides a mailbox name and password, the Send Note dialog box appears.

The user's default messaging options are used as the default dialog box values. The function caller is responsible for deleting temporary files created when using this function.

Parameter	Description
UIParam	The parent window handle for the dialog box. A value of 0 specifies that the Send Note dialog box is application modal.
DelimChar	A string containing the character used to delimit the names in the *FilePaths* and *FileNames* parameters. This character should not be used in filenames on your operating system.
FilePaths	A string containing the list of full paths (including drive letters) for the attached files. The list is formed by concatenating correctly formed file paths separated by the character specified in the *DelimChar*

	parameter. An example for a Windows or MS-DOS system is: `C:\TMP\TEMP1.DOC;C:\TMP\TEMP2.DOC` The files specified in this parameter are added to the message as file attachments. If this parameter contains an empty string, the Send Note dialog box is displayed with no attached files.
FileNames	A string containing the list of the original filenames (in 8.3 format) as they should be displayed in the message. When multiple names are specified, the list is formed by concatenating the filenames separated by the character specified in the *DelimChar* parameter. An example for a Windows or MS-DOS system is: `MEMO.DOC;EXPENSES.DOC` Note that the icon displayed for a file is based on the filename extension supplied in this parameter. For example, a filename with an .XLS extension is displayed with a Microsoft Excel icon. The messaging system also relies on the file extension when opening and saving a file. If an attached file has no extension, append the default extension for your application's document type.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	**Meaning**
MP_E_ATT_OPEN_FAILURE	One or more files in the *FilePaths* parameter could not be located. No mail was sent.
MP_E_DISK_FULL	The disk was full.
MP_E_FAILURE	One or more unspecified errors occurred while sending the mail. It is not known if the mail was sent.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed.
MP_E_LOGIN_FAILURE	There was no default sign in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No mail was sent.
MP_USER_ABORT	The user canceled the process (from the Send Note dialog box). No mail

	was sent.
SUCCESS_SUCCESS	The mail was successfully sent. The caller is responsible for deleting any temporary files referenced in the *FilePaths* parameter.

MPSendMail

{bmc retrnval.bmp}{bmc seealso.bmp}

```
MPSendMail(
    Session as Numeric,
    UIParam as Numeric,
    Message as MapiMesg,
    Recips as MapiRecip,
    Files as MapiFile,
    Flags as Numeric,
    Reserved as Numeric) as Numeric
```

Description

This function sends a standard mail message. If you choose, it can prompt for user input with a dialog box or proceed without any user interaction.

You can optionally provide a list of recipient names, subject text, file attachments, or message text when you call **MPSendMail**. If you do not supply the required message elements, the function can prompt the user for them. If you provide recipient names, file attachments, or message text, the function can send the files or note without prompting users. If the optional parameters are specified and a dialog box is requested by use of the MP_DIALOG flag, the parameters provide the initial values for the dialog box.

File attachments are copied to the message before the function returns; therefore, later changes to the files do not affect the contents of the message. The files must be closed when they are copied.

The FoxPro **MPSendMail** function takes the recipients and file attachments from the appropriate parameters to the call. *Recips* and *Files* should each be the first element of dynamically allocated arrays of their respective types. These arrays are not redimensioned and are not affected by assignment side-effects.

All strings are to be specified in the current character set or code page of the calling program's operating system process. In Microsoft

Windows and MS-DOS environments, the character set is ANSI. In OS/2 environments, the code page is CP-850.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. If the value is 0, the messaging system sets up a session either from a system default session (if one exists) or presents a sign-in dialog box. In all cases, the messaging system returns to its state before the call.
UIParam	The parent window handle for the dialog box. A value of 0 specifies that any dialog box displayed is application modal.
Message	**Subject**
	An empty string indicates no subject text. Some implementations may truncate subject lines that are too long or contain carriage returns, line feeds, or other control characters.
	Note Text
	An empty string indicates no text. Each paragraph should be terminated with either a carriage return (0x0d), a line feed (0x0a), or a carriage return-line feed pair (0x0d0a). The implementation wraps lines as appropriate. Implementations may place limits on the size of the text. A return of MP_E_TEXT_TOO_LARGE is generated if this limit is exceeded.
Message Type	
	A pointer to a string that is the message type. This field is for use by applications other than interpersonal mail (electronic forms, game message transmittal, and so on). For an interpersonal mail message, specify an empty string for this field.
Recips	The first element of an array of recipients. When *RecipCount* is zero, this parameter is ignored.
	The recipient string can include either the recipient's name or the recipient's name-address pair. If just a name is specified, the name is resolved to an address using implementation-defined address book search rules. If an address is also specified, a search for the name is not performed. The address is in an implementation-defined format and is assumed to have been obtained from the implementation in some

other way. When the address is specified, the name is used for display to the user and the address is used for delivery.

When *EntryID* is used, no search is performed and the displayed name and address are ignored. (A name and address are associated with *EntryID* within the messaging system.) *EntryID*s are returned by the **MPReadMail** function.

Files

The first element of an array of attachment files written when the message is read. The number of attachments per message may be limited in some systems. If the limit is exceeded, the error MP_E_TM_FILES is returned. When *FileCount* is 0, this parameter is ignored.

Attachment files are read and attached to the message before the call returns. Do not attempt to display attachments outside the range of the message body.

Flags

A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG:

```
#define MP_LOGON_UI   1
' Display logon UI
#define MP_NEW_SESSION   2
' Don't get default if available
#define MP_DIALOG   8
' Display a send note UI
```

Set MP_LOGON_UI if the function should display a dialog box to prompt for sign in (if required). When this flag is not set, the function does not display a dialog box and returns an error if the user is not signed in.

Set MP_NEW_SESSION if you want to establish a session other than a current session. For instance, if a mail client is already running, another MAPI electronic-mail client can piggyback on the session created by the mail client application. Do not set this flag if you want the default session (if it still exists). If the session passed in *Session* is not 0, this flag is ignored.

Set MP_DIALOG if the function should display a dialog box to prompt for recipients and other sending options. When this flag is not set, the function does

	not display a dialog box, but at least one recipient must be specified.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	Meaning
MP_E_AMB_RECIP	A recipient matched more than one of the recipient descriptor types, and MP_DIALOG was not set. No mail was sent.
MP_E_ATT_	
NOT_FOUND	The specified attachment was not found. No mail was sent.
MP_E_BAD_RECIPTYPE	The type of a recipient was not MP_TO, MP_CC, or MP_BCC. No mail was sent.
MP_E_DISK_FULL	The disk was full. No mail was sent.
MP_E_FAILURE	One or more unspecified errors occurred while sending the mail. No mail was sent.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No mail was sent.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No mail was sent.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No mail was sent.
MP_E_TEXT_TOO_LARGE	The text in the message was too large to be sent. No mail was sent.
MP_E_TM_FILES	There were too many file attachments. No mail was sent.
MP_E_TM_RECIPIENTS	There were too many message recipients specified. No mail was sent.
MP_E_TM_SESSIONS	The user had too many sessions open at once. No mail was sent.
MP_E_UNKNOWN_RECIPIENT	The recipient did not appear in the address list. No mail was sent.
MP_USER_ABORT	The user canceled the process from

	the send dialog box. No mail was sent.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPReadMail

MPCursor

{bmc retrnval.bmp}

```
MPCursor(
    MPCursor as String,
    AliasName as String) as Numeric,
```

Description

In some cases it is necessary to create cursors before calling one of the FoxMAPI functions. The MPCursor function is used to create the MAPI cursors as desired. The function can take either one or two parameters. The first parameter is the name of the FoxMAPI cursor to create, where these names correspond to those given in the tables above. The second parameter is the alias name used for refer to the cursor. If the alias name is omitted then the name of the FoxMAPI cursor is used. i.e. in the first example below an instance of the SPlusRest cursor is created with an alias name of SPlusRest:

=MPCursor("SPlusRest", "SPlusRestriction")

In the second example, an instance of the MapiRecip cursor is created with an alias name of MapiRecip:

=MPCursor("MapiRecip")

NOTE: If an alias of the same name is already open in a workarea, the MPCursor function will simply leave that table intact as it exists. You will need to check for this before making a call to MPCursor, especially if you want an empty cursor.

Parameter	Description
MPCursor	The name of the FoxMAPI cursor to create, where these names correspond to one of the following (MapiMesg, MapiRecip,MapiFile).
AliasName	The alias name used to refer to the cursor. If the alias name is omitted then the name of the FoxMAPI cursor is used.

Return Value	Meaning
SUCCESS_SUCCESS	The function returned successfully.

SPlusAppt

{bmc seealso.bmp}

```
Cursor SPlusAppt
    Reserved As Numeric
    ItemType As String
    Flags As Numeric
    AssocCount As Numeric
    OrgItemID As String
    AttendCnt As Numeric
    Text As String
    Body As String
    Recurrence as String
    DateStart As String
    DateEnd As String
```

Description

Schedule+ Libraries allow four kinds of appointments:

{bmc b.bmp} **SimpleAppt** stores a unique event on a user's calendar.
{bmc b.bmp} **RecurApptInstance** stores an exception to a recurring event on a user's calendar.
{bmc b.bmp} **OrganizedMeeting** stores a meeting on a meeting organizer's calendar.
{bmc b.bmp} **BookedMeeting** stores a meeting on an attendee's calendar.

The **SPlusAppt** type specifies the kind of appointment and specifics related to that appointment.

Field	Description	Type
Reserved	Reserved for future use. This field must be 0.	N(10,0)
ItemType	A string identifying the appointment type. This field must be one of the following strings: **SimpleAppt**, **RecurApptInstance**, **OrganizedMeeting**, or	C(254)

	BookedMeeting.	
Flags	A bitmask of flags. Unused flags are reserved. The following flags are supported and defined in MAPILIB.PRG: #define SP_PRIVATE 1 #define SP_TENTATIVE 4	N(10,0)
AssocCount	The number of items associated with this appointment. In this release, the only kind of associated item allowed is an alarm, and only one alarm is allowed. Therefore, this value can be only 0 or 1. However, to be compatible with future releases, your code should allow for more than one associated item.	N(10,0)
OrgItemID	A string specifying the Schedule+ ID for the referenced message on the organizer's schedule. This field is ignored if *ItemType* is not **OrganizedMeeting** or **BookedMeeting**.	C(254)
AttendCnt	Number of attendees for a meeting. This field must be 0 if *ItemType* is not **OrganizedMeeting**. This field must not be 0 if it is **OrganizedMeeting**.	N(10,0)
Text	The text field of the appointment. An empty string indicates no text.	M
Body	Reserved for future use. This field must be an empty string.	M
Recurrence	Reserved for future use. This field must be an empty string.	C(254)
DateStart	A string indicating the start date and time of the meeting. The format is YYYY/MM/DD HH:MM; hours are measured on a 24-hour clock. The year must be in the range 1920-2019.	C(16)
DateEnd	A string indicating the end date and time of the meeting. The format is YYYY/MM/DD HH:MM; hours are measured on a 24-hour clock. The year must be in the range 1920-2019. The	C(16)

end date must be later than the start date, and the elapsed time cannot exceed 31 days.

See Also

SPReadAppt

SPSaveAppt

SPlusAssoc

```
Cursor SPlusAssoc
    Reserved As Numeric
    ItemType As String
    Flags As Numeric
    ItemID As String
    Position As Numeric
    Data As String
```

Description

In this release, alarms are the only type of associated items allowed.

Field	Description	Type
Reserved	Reserved for future use. This field must be 0.	N(10,0)
ItemType	In this release, this field must be **Alarm**.	C(254)
Flags	Reserved for future use. This field must be 0.	N(10,0)
ItemID	A pointer to a Schedule+ internal ID for this item. If you are creating a new associated item with **SPSaveAppt** or **SPSaveTask**, pass in an empty, zero-terminated, caller-allocated string long enough to accommodate at least 64 characters.	C(254)
Position	Reserved for future use. This field must be 0.	N(10,0)
Data	A string providing data specific to the item type. For items of type **Alarm**, this string indicates the date and time	C(254)

when the alarm should ring. The format is YYYY/MM/DD HH:MM; hours are measured on a 24-hour clock. The year must be in the range 1920-2019.

SPlusAttd

```
Cursor SPlusAttd
    Reserved As Numeric
    UserType As String
    Status As Numeric
```

Field	**Description**	**Type**
Reserved	Reserved for future use. This field must be 0.	N(10,0)
UserType	A string specifying the user type. For this release, the two supported types are **Individual** and **Unspecified Resource**.	C(254)
Status	Specifies the current confirmation status of the attendee. This field must be one of the following: #define SP_NO_REQUEST_SENT 0 #define SP_NO_RESPONSE 1 #define SP_POSITIVE_RESPONSE 2 #define SP_NEGATIVE_RESPONSE 3 #define SP_AMBIGUOUS_RESPONSE 4	N(10,0)

FoxPro Environment Considerations

{bmc seealso.bmp}

Where to Install DLL and FLL Files

The three DLLs, MAPI.DLL, SPLUS.DLL, and EFORM.DLL must be installed in your windows\system directory since they are loaded and called by native Windows functions. The FOXMAPI.FLL library, on the other hand, can reside anywhere the user can access it through the FoxPro SET LIBRARY TO command.

Additional FoxPro Files

Included with the FoxPro MAPI libraries are two FoxPro generic programs (MAPILIB.PRG and MAPIERR.PRG) which can be used to handle all calls to the libraries. The MAPILIB.PRG file can and should be modified as needed to support specific applications. In its present form, it is only intended to provide routines for some of the most common uses of MAPI. All calls in the FOXMAPI.FLL library have return values. These return values/error codes are contained in the MAPIERR.PRG file as identified as #DEFINES.

FoxPro Cursors

Many of the MAPI calls create and/or make use of FoxPro cursors (temporary DBFs). There are three specific ones. The **MailMesg** cursor contains information about a specific mail message. There is only one record created in the cursor. The **MailFile** cursor contains a list of all of the attachments (i.e., files/ole objects) associated with a particular message. The **MapiRecips** cursor has a record for each recipient of the specific message. NOTE: a **MapiOrig** cursor with the same structure as the MapiRecips cursor is also created with a single record of the message originator. Be aware of the 1-Many relations associated with the above cursors. You can use the MPCursor call to create a cursor for any of these 3 cursor types.

The cursors can be used like ordinary DBFs for most database operations such as INDEX, ZAP, APPEND BLANK, INSERT, etc. NOTE: the field lengths of many character fields are the maximum permissible length of 254. This is done primarily to accommodate data received from the mail messages. If you are only using MAPI to send messages, you can create you own cursors containing the same field names, but with shorter field lengths. The functions will still work the same.

Other Considerations

There are several issues the programmer should be aware of with the MPReadMail function. If you are making subsequent calls to this function, you must check the MapiMesg.Filecount field to see if it is zero. If the value is zero, the MapiFile cursor would be unaffected and could contain data from the previous MPReadMail call. Another consideration is that file attachments are created in your Windows temp directory. You will need to properly handle/dispose of them because each subsequent MPReadMail call to the same message will result in a duplicate file being created in this directory.

You can create Mail messages which are not detected by the MS Mail client. These are known as IPC messages and can be created using the MessageID field in the MapiMesg cursor. IPM type messages, which are the default type, are the only ones detected by the MS Mail client. NOTE: The IPC string should be unique (ie IPC.MYAPP).

You do not need to call the MPLogon function to start a mail session prior to making FoxPro MAPI calls. By including a Session value of zero and a Flag bitmask value of 1 (MP_LOGON_UI), the logon dialog will become available. Be aware, however, that you will be automatically logged out of Mail after this call is made should you decide to take this approach. If you are running a program involving many MAPI calls, then it is best to use the MPLogon call and save the Session handle for use with subsequent MAPI calls.

The cursors created from a MPReadMail call do not contain a common field which could be used to relate the databases. If you decide that you want to log incoming messages into a database along with their origin, recipient, and file attachment information, you will need to create new databases which have a unique key field. The MapiMesg cursor does contain a unique message ID field, however, the other cursors have no reference to this.

The RecipClass field in the MapiRecips cursor refer to the class of the recipient.

0 - Originator; 1 - To; 2 - CC; 3 - BCC

A quick way to give the user functionality to send a message is by using the following command:

```
retVal=MPSendDocs(0,";","","",0)
```

The MPSendDocs function always presents an MS Mail SendNote dialog since you cannot provide information regarding the recipients. If you want to programmatically control who receives the message, use the MPSendMail function. It is much more powerful.

See Also

MAPI Library

Schedule+ Library

SPlusMesg

{bmc seealso.bmp}

```
Cursor SPlusMesg
    Reserved As Numeric
    Subject As String
    NoteText As String
    MessageType As String
    DateReceived As String
    ConversationID As String
    Flags As Numeric
    RecipCount As Numeric
    FileCount As Numeric
    SentForCount As Numeric
```

Description

The **SPlusMesg** type contains information about a Schedule+ meeting request. The initial portion of this type is the same as the **MapiMesg** type.

Field	Description	Type
Reserved	Reserved for future use. This field must be 0.	N(10,0)
Subject	A subject text field string, limited to 256 characters or less. An empty string indicates no subject text.	C(254)
NoteText	A string containing text in the message. An empty string indicates no text. For inbound messages, each paragraph is terminated with a carriage return–line feed pair (0x0d0a). For outbound messages, paragraphs can be delimited with a carriage return, a line feed, or a carriage return–line feed pair (0x0d, 0x0a, or 0x0d0a).	M
MessageType	Must be one of the following strings: "IPM.Microsoft Schedule.MtgReq" "IPM.Microsoft Schedule.MtgRespP" "IPM.Microsoft Schedule.MtgRespN" "IPM.Microsoft Schedule.MtgRespA" "IPM.Microsoft Schedule.MtgCncl"	C(254)

	The strings represent a meeting request, a positive meeting response, a negative meeting response, a tentative meeting response, and a meeting cancellation, respectively.	
DateReceived	A string indicating the date a message is received. The format is YYYY/MM/DD HH:MM; hours are measured on a 24-hour clock.	C(16)
ConversationID	A string indicating the conversation thread ID to which this message belongs.	C(254)
Flags	A bitmask of flags. Unused flags are reserved. Unused flags must be 0 for outbound messages and are ignored for inbound messages. The following flags are defined in MAPILIB. PRG: #define MP_UNREAD ONLY 1 #define MP_SENT 4 The flag MP_RETURN_RECEIPT_REQUESTED is not supported in this release. An additional flag defined in MAPILIB.PRG is also supported: #define SP_RESPONSE_REQUESTED 65536 This flag is ignored for outgoing responses and cancellations.	N(10,0)
RecipCount	A count of the recipient descriptor types pointed to by *Recips.* A value of 0 indicates that no recipients are included.	N(10,0)
FileCount	Meeting messages cannot have file attachments in this release. This field must be 0 when using **SPSendMeet**. Note that it will be 1 when *Flags* in **SPReadMeet** is set to MP_BODY_AS_FILE.	N(10,0)
SentForCount	A count of the recipient descriptor types pointed to by *SentFor*. A value of 0 indicates no recipient descriptor types. This value must be 0 for outgoing meeting requests and	N(10,0)

cancellations. This value must be 1 for outgoing meeting responses.

See Also

MapiMesg

SPReadMeet

SPSendMeet

SPlusRest

{bmc seealso.bmp}

```
Cursor SPlusRest
    Reserved As Numeric
    ItemType As String
    RestType As String
    RestData As String
```

Description

Depending on the item type (task or appointment), **SPlusRest** fetches only appointments for a specified day, only tasks for a specified day, or all tasks.

Field	Description	Type
Reserved	Reserved for future use. This field must be 0.	N(10,0)
ItemType	This field must be either **Appt** or **Task**.	C(254)
RestType	If the item type is **Appt**, *RestType* must be **Day**. If the item type is **Task**, *RestType* must be either **All** or **Active.**	C(254)
RestData	For restrictions on appointments, this parameter is a string indicating the day of the appointment. The format is YYYY/MM/DD. The year must be in the range 1920-2019. An empty string specifies the current day. For restrictions on tasks, this parameter is a string indicating the date of the task. The format is YYYY/MM/DD HH:MM; hours are measured on a 24-hour clock. The year must be in the range	C(254)

1920-2019. An empty string specifies the current date and time.

See Also

SPReadAppt

SPReadTask

SPlusTask

```
Cursor SPlusTask
    Reserved As Numeric
    ItemType As String
    Flags As Numeric
    AssocCount As Numeric
    OrgItemID As String
    AttendCnt As Numeric
    Text As String
    Body As String
    Recurrence As String
    DateDue As String
    DuratActiv As String
    ProjctName As String
    Priority As String
```

Field	**Description**	**Type**
Reserved	Reserved for future use. This field must be 0.	N(10,0)
ItemType	Identifies the item type. This field must be **SimpleTask** or **RecurTaskInstance**.	C(254)
Flags	A bitmask of flags. Unused flags are reserved. The following flags are defined in MAPILIB.PRG : #define SP_PRIVATE 1 #define SP_PRV_PROJECT 2	N(10,0)
AssocCount	The number of items associated with this appointment. In this release, the only kind of associated item allowed is an alarm, and only one alarm is allowed. Therefore, this value can be only 0 or 1. However, to be compatible	N(10,0)

	with future releases, your code should allow for more than one associated item.	
OrgItemID	Reserved for future use.	C(254)
AttendCnt	Reserved for future use.	N(10,0)
Text	The text field for the task. An empty string indicates no text.	M
Body	Reserved for future use. This field must be an empty string.	M
Recurrence	Reserved for future use. This field must be an empty string.	C(254)
DateDue	A string indicating the due date for the task. The format is YYYY/MM/DD HH:MM; hours are measured on a 24-hour clock. The year must be in the range 1920-2019. If the task doesn't have a due date, this string should be an empty string.	C(16)
DuratActiv	A string indicating how soon a task is active before the due date. This string is ignored if the task has no due date. The duration can be specified as a number of days, weeks, months, or years. The format is NNNN U, where NNNN is the number of units, and U is either **D** for days, **W** for weeks, or **M** for months.	C(6)
ProjctName	A string indicating the name of the parent project for a task. An empty string indicates that there is no parent project. The flag SP_PRV_PROJECT indicates whether the parent project is private or not.	C(254)
Priority	A string indicating the priority of a task. The priority value should be one character, either a digit from 1 - 9 or an uppercase letter from A - Z. An empty string indicates a default value.	C(1)

SPlusUser

```
Cursor SPlusUser
    Reserved As Numeric
    UserType As String
    Flags As Numeric
    StartTime As String
    EndTime As String
    TimeZone As String
    Data As String
```

Field	**Description**	**Type**
Reserved	Reserved for future use. This field should be ignored.	N(10,0)
UserType	A string specifying the user type. For this release, the two supported types are **Individual** and **Unspecified Resource**.	C(254)
Flags	A bitmask of flags. Unused flags are reserved. The following flag is defined in MAPILIB.PRG: #define SP_BOSS_WANTS_COPY 1	N(10,0)
StartTime	Indicates the start of working hours for the logged in user. The format is HH:MM, where MM must be either **00** or **30**.	C(254)
EndTime	Indicates the end of working hours for the logged in user. The format is HH:MM, where MM must be either **00** or **30**. The end time value must be later than the start time.	C(254)
TimeZone	Reserved for future use. This field should be ignored for this release.	C(254)
Data	Reserved for future use. This field should be ignored for this release.	C(254)

SPReadMeet

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPReadMeet(
    Session As Numeric,
    UIParam As Numeric,
    MessageID As String,
    Flags As Numeric,
    Reserved As Numeric,
    Message As SPlusMesg,
    Originator As MapiRecip,
    Recips() As MapiRecip,
    Files() As MapiFile,
    SentFor() As MapiRecip,
    Appt As SPlusAppt,
    Assoc() As SPlusAssoc,
    Attendees() As SPlusAttd
    UserType As String
) as Numeric
```

Description

Before calling this function, use **MAPILogon** to establish a valid MAPI session, and then use **MAPIFindNext,** specifying one of the message types listed in the description of the *MessageType* field of **SPlusMesg**, before calling **SPReadMeet.**

The call returns one message, filling the **SPlusMesg** data type with the elements of the message. Recipients and contents are copied from the message before the function returns to the caller, so later changes do not affect the contents of this message.

When the message is a meeting request or meeting cancellation, the *SentFor* field provides a list of users for which the request is intended. Your application must then act on the meeting request or cancellation for each of these users.

All strings are null-terminated and must be specified in the ANSI character set.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. If the value is 0, **SPReadMeet** returns the error MP_E_INV_SESSION.

UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
MessageID	A caller-allocated string that is the message identifier of the message to be read. Message IDs can be obtained from the **MAPIFindNext** and **SPLUSSaveMeeting** functions.
Flags	A bitmask of flags. Unspecified flags should always be set to 0. Undocumented flags are reserved. The following flags are defined in MAPILIB.PRG: #define MP_ENVELOPE_ONLY 64 ' Only header information #define MP_SUPPRESS_ATTACH 2048 ' Header and body, no files #define MP_BODY_AS_FILE 512 ' Save body as first attachment #define MP_PEEK 128 ' Don't mark message as read When MP_ENVELOPE_ONLY is set, the function does not copy file attachments to temporary files or return the note text. All other message information (except for temporary file names) is returned. Setting this flag usually reduces the processing time required for the function. When MP_SUPPRESS_ATTACH is set, **SPReadMeet** does not copy file attachments, but returns message text. This flag is ignored if MP_ENVELOPE_ONLY is set. The flag should reduce the time required by the **SPReadMeet** function. When MP_BODY_AS_FILE is set, **SPReadMeet** writes the message body to a temporary file and adds it to the attachment list as the first attachment, instead of returning a pointer to the message body (the default behavior). When MP_PEEK is set, **SPReadMeet** does not mark the message as read. Any unsuccessful return leaves the message unread.
Reserved	Reserved for future use. This parameter must be 0.
Message	A message type containing the message contents. When **SPReadMeet** is called with MP_BODY_AS_FILE set, you should delete these files when they are no longer needed.
Originator	A **MapiRecip** recipient descriptor type that describes the originator of the meeting request.
Recips ( )	A dynamic array of recipient descriptor types. Only recipients of type **MP_TO** are supported in this

	release.
Files ( )	If Flags is set to MP_BODY_AS_FILE, then a struct of type *MapiRecip* is returned, otherwise it is set to empty string.
SentFor ( )	A dynamic array of recipient descriptor types. On an incoming meeting request or cancellation, this parameter gives a list of users (recipient descriptors of type **MP_TO**) for which the request is intended. Your application must act on the meeting request or cancellation for each of these users. A meeting response should identify the user that the sender is responding to.
Appt	An **SPlusAppt** type with a **BookedMeeting** *ItemType*. In this release, the fields of the appointment type are set as follows. (These are subject to change in later releases.) *Flags* This field will be set to 0. *Creator* This field will be set to a recipient descriptor with all fields equal to 0 or empty strings (as appropriate). *AssocCount* This field will be set to 0. *Organizer* This field will be set to the identity of the meeting organizer. *OrgItemID* This field is set depending on the sender, and may be an empty string. If this field does not point to an actual item ID, it indicates that the sender is not a Schedule+ user. *AttendCnt* This field will be set to 0. *Text* This field will be set the same as the message subject text. *Body* and *Recurrence* These fields will be set to an empty string. *DateStart, DateEnd* These fields will be set appropriately.
Assoc ( )	This field will be ignored.
Attendees ( )	This field will be ignored.
UserType	A string specifying the user type. For this release, the two supported types are **Individual** and **Unspecified**

Resource.

Return Value	**Meaning**
MP_E_ATT_WRITE_FAILURE	The message body could not be written to a temporary file (when MP_BODY_AS_FILE is used). Check directory permissions.
MP_E_DISK_FULL	The disk was full.
MP_E_FAILURE	Unspecified error(s) occurred while reading the message. No message was read.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No message was read.
MP_E_INV_MESSAGE	The message ID was invalid. The message ID may have been deleted or changed by another process.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No message was read.
MP_E_NOT_SUPPORTED	The operation was not supported by the underlying messaging system.
MP_E_TM_RECIPIENTS	There were too many recipients of the message. No message was read.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MapiRecip

SPlusAppt

SPSendMeet

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPSendMeet(
    Session As Numeric,
    UIParam As Numeric,
    Message As SPlusMesg,
    Recips() As MapiRecip,
    Files() As MapiFile,
    SentFor() As MapiRecip,
    Appt As SPlusAppt,
    Assoc() As SPlusAssoc,
    Attendees() As SPlusAttd,
    UserType As String
    Flags As Numeric
    Reserved As Numeric
 ) as Numeric
```

Description

You provide a list of recipient names, subject text, message text, and meeting information when you call **SPSendMeet**. Unlike **MPSendMail**, if you do not supply the required message elements, the function fails instead of prompting the user for the missing elements.

All strings are null-terminated and must be specified in the ANSI character set.

Parameter	Description
Session	An opaque session handle whose value represents a session with the messaging system. If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. In all cases, this function returns with the sign-in state unchanged.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
Message	An **SPlusMesg** type containing the contents of the message to be sent. Set the following fields for successful message delivery: *SPlusMesg.MessageType* *SPlusMesg.RecipCount*

Meeting responses (only) should set the following fields:

SPlusMesg.SentForCount = 1
SPlusMesg.SentFor

All other fields are optional. Unused fields should be 0 or empty strings. The following fields are ignored and should be 0 or empty strings:

SPlusMesg.DateReceived
SPlusMesg.Originator
SPlusMesg.Flags
SPlusMesg.FileCount
SPlusMesg.File

Subject
An empty string indicates no subject text. Some implementations may truncate subject lines that are too long or contain carriage returns, line feeds, or other control characters.

NoteText
An empty string indicates no text. Each paragraph must be terminated with either a carriage return (0x0d), a line feed (0x0a), or a carriage return-line feed pair (0x0d0a). Different implementations can wrap lines and place limits on the size of the text, as appropriate. If the limit is exceeded, a return of MP_E_TEXT_TOO_LARGE is generated.

MessageType
This field must be one of the following strings*:*

```
"IPM.Microsoft Schedule.MtgReq"
"IPM.Microsoft Schedule.MtgRespP"
"IPM.Microsoft Schedule.MtgRespN"
"IPM.Microsoft Schedule.MtgRespA"
"IPM.Microsoft Schedule.MtgCncl"
```

These strings represent a meeting request, a positive meeting response, a negative meeting response, a tentative meeting response, and a meeting cancellation, respectively.

FileCount
Attachments are not supported by this release. The count of files should be 0.

Recips ( ) An array of recipient descriptor types.

Recipients *(Recips)*
The recipient descriptor can include either the recipient’s name or the recipient’s name-address pair.

	If only a name is specified, the name is resolved to an address using implementation defined address-book search rules. If an address is also specified, a search for the name is not performed. The address is in an implementation defined format and is assumed to have been obtained from the implementation some other way. When the address is specified, the name is displayed to the user and the address is used for delivery.
	When *EntryID* is specified, no search is performed and the displayed and address are ignored. (A name and address are associated with *EntryID* in the messaging system.) *EntryID*s are returned by the **SPReadMeet** function.
Files ( )	Meeting messages cannot have file attachments in this release.
SentFor ( )	For a meeting request this field is ignored. For a meeting response this field identifies the user that the sender is responding to.
Appt	An **SPlusAppt** type containing information about the meeting.
	For meeting responses, the appointment type should be the same as the one received with the original meeting request.
	For meeting requests and cancellations, the appointment type must be **OrganizedMeeting** and must be initialized properly.
Assoc ( )	An array of associated item descriptors. In this release, the only type of associated items allowed are alarms.
Attendees ( )	An array of attendee descriptors specifying the meeting's attendees and their confirmation status.
UserType	A string specifying the user type. For this release, the two supported types are **Individual** and **Unspecified Resource**.
Flags	Reserved for future use. This parameter must be 0.
Reserved ( )	Reserved for future use. This parameter must be 0.

Return Value	**Meaning**
MP_E_AMB_RECIP	A recipient matched more than one of the recipient descriptor types.

MP_E_BAD_RECIPTYPE	The type of a recipient was not **MP_TO**, **MP_CC**, or **MP_BCC**.
MP_E_DISK_FULL	The disk was full.
MP_E_FAILURE	Unspecified error(s) occurred while sending the message. No message was sent.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No message was sent.
MP_E_INV_SESSION	An invalid session handle was used for the *Session* parameter. No message was sent.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No message was sent.
MP_E_TEXT_TOO_LARGE	The text in the message was too large to be sent. No message was sent.
MP_E_TM_FILES	A file attachment was specified. No message was sent.
MP_E_TM_RECIPIENTS	There were too many message recipients specified. No message was sent.
MP_E_TM_SESSIONS	The user had too many sessions open at once. No message was sent.
MP_E_UNKNOWN_RECIPIENT	The recipient did not appear in the address list. No message was sent.
MP_USER_ABORT	The user canceled the process. No message was sent.
SP_E_TIME	The *DateStart* or *DateEnd* field of the **SPlusAppt** type was not specified properly.
SP_E_INV_SENT_FOR	The *SentFor* parameter was not specified properly.

SP_E_TYPE	An invalid item type was supplied in the **SPlusAppt** type.
SP_E_ASSOC	There was an error in the way the associated items were specified.
SP_E_ORGANIZER	The *Organizer* field of the **SPlusAppt** type was not specified properly.
SP_E_ORG_ID	The *OrgItemID* field of the **SPlusAppt** type was not specified properly.
SP_E_ATTENDEES	The associated appointment does not have attendees.
SP_E_FLAGS	The *Flags* field of the **SPlusAppt** type was not specified properly.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPSendMail

SPlusAppt

SPlusMesg

SPReadMeet

SPBegin

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPBegin(
    MapiSession As Numeric
    UIParam As Numeric
    Flags As Numeric
    Reserved As Numeric
    SPlusSession As Reference to a Numeric
 ) as Numeric
```

Description

SPBegin returns a session handle necessary for other Schedule+ Libraries calls. To make a series of Schedule+ Libraries calls, you should begin a Schedule+ Libraries session with **SPBegin**, make the necessary calls, and then terminate the Schedule+ Libraries session with **SPEnd.**

Parameter	Description
MapiSession	An opaque session handle whose value represents an existing session with the messaging system. Session handles are returned by **MPLogon** and invalidated by **MPLogoff.** If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
Flags	Reserved for future use. This parameter must be 0.
Reserved	Reserved for future use. This parameter must be 0.
SPlusSession	An opaque session handle whose value is set by Schedule+ when the **SPBegin** call is successful. The session handle can then be used in subsequent Schedule+ Libraries calls.

Return Value	Meaning
MP_E_FAILURE	Unspecified error(s) occurred while attempting this call. No session handle was returned.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No session handle was returned.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No session handle was returned.
MP_E_TM_SESSIONS	The user had too many sessions open at once. No session handle was returned.
MP_USER_ABORT	The user canceled the process. No session handle was returned.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved*

	parameter was not set correctly. No session handle was returned.
SP_E_NOT_INSTALLED	Schedule+ was not installed.
SP_E_NO_SCHEDULE	No schedule file was found for the user. The user should stop and restart Schedule+.
SUCCESS_SUCCESS	The function returned successfully.

See Also

SPEnd

SPDelete

{bmc retrnval.bmp}

```
SPDelete(
    SPlusSession As Numeric
    UIParam As Numeric
    SPlusUser As MapiRecip
    ItemID As String
    Flags As Numeric
    Reserved As Numeric
 ) as Numeric
```

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd.** If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. Otherwise, calls with *SPlusSession* equal to 0 return MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
SPlusUser	A recipient descriptor specifying the user. The *Name* and *Address* field must be completed. A descriptor with all fields equal to 0 or empty strings (as appropriate) indicates that you want to delete the item from the schedule of the currently signed-in user.

ItemID	The string identifier for the item to be deleted. This identifier is invalid after **SPDelete** returns successfully.
Flags	A bitmask of flags. All flags are reserved and should be set to 0.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	**Meaning**
MP_E_DISK_FULL	The disk was full. No changes were made.
MP_E_FAILURE	Unspecified error(s) occurred while saving the mail. Nothing was deleted.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No changes were made.
MP_E_INV_SESSION	An invalid session handle was used for *SPlusSession* parameter. No changes were made.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No changes were made.
MP_USER_ABORT	The user canceled the process. No changes were made.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. No changes were made.
SP_E_USER	The recipient descriptor was incomplete or specified incorrectly. No changes were made.
SP_E_NO_SCHEDULE	No schedule was located for the user. No changes were made.
SP_E_ITEM	The item ID was invalid. The item ID may have been deleted by another process. No changes were made.

SP_E_NOT_INSTALLED	Schedule+ was not installed.
SP_E_NO_PRIVILEGE	The caller did not have a privilege level of **Create** or greater for the user's schedule.
SUCCESS_SUCCESS	The function returned successfully.

SPEnd

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPEnd(
    SPlusSession As Numeric
    UIParam As Numeric
    Flags As Numeric
    Reserved As Numeric
 ) as Numeric
```

Description

Ending unnecessary sessions helps avoid the MP_E_TM_SESSIONS error with other Schedule+ Libraries function calls.

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd**.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
Flags	Reserved for future use. This parameter must be 0.
Reserved	Reserved for future use. This parameter must be 0.

Return Value	Meaning
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. The session was not terminated.
MP_E_INV_SESSION	An invalid session handle was used for the *SPlusSession* parameter. The session was not terminated.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved*

	parameter was not set correctly. The session was not terminated.
SUCCESS_SUCCESS	The function returned successfully.

See Also

SPBegin

SPFindNext

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPFindNext(
    SPlusSession As Numeric
    UIParam As Numeric
    SPlusUser As MapiRecip
    Restriction As SPlusRest
    SeedItemID As String
    Flags As Numeric
    Reserved As Numeric
    ItemID As Reference to a String
 ) as Numeric
```

Description

This function allows an application to enumerate appointments and tasks. Successive calls to this function can read all appointments on a specified day of a user's schedule, or all of the user's tasks.

When provided with an empty string in *SeedItemID,* **SPFindNext** returns the ID and type of the first item on the schedule of *SPlusUser* subject to the value of *Restriction.* When provided with a non-empty *SeedItemID*, **SPFindNext** returns the next matching item. Repeated calls to **SPFindNext** ultimately result in a return of SP_E_NO_ITEMS, which means that the enumeration of the matching items is complete.

Item identifiers are not guaranteed to remain valid, because other applications could delete items between calls. Applications must be able to handle failed calls to **SPFindNext, SPDelete, SPReadAppt,** and **SPReadTask** for invalid item IDs. The ordering of items is system-specific. Item ID strings must be allocated by the caller and must be at least 64 characters long.

SPFindNext handles item enumeration in the following way:

 For recurring, single appointment types, or tasks **SPFindNext** returns the next appointment as qualified by the *Restriction* parameter.

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd.** If the value is 0, this function returns MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
SPlusUser	A recipient descriptor specifying the user. The *Name* and *Address* fields must be completed. A descriptor with all fields 0 or empty strings (as appropriate) indicates that you want to read information for the currently signed-in user.
Restriction	A restriction descriptor that indicates the type of item to read.
SeedItemID	A string that is the item identifier seed for the request. If the identifier is an empty string, the first item matching the restriction is returned. Item IDs are system-specific and opaque. Item IDs might be invalidated at any time if another application deletes an item. A valid SeedItemID can be obtained by using SPFindNext.
Flags	A bitmask of flags. All flags are reserved and should be set to 0.
Reserved	Reserved for future use. This parameter must be 0.
ItemID	A caller-allocated string that is the item identifier. Item identifiers are system specific, may be non printable, and are opaque. Item ID strings must be able to accommodate at least 64 characters. Item IDs might be invalidated at any time if another application deletes an item.

Return Value	Meaning
MP_E_FAILURE	Unspecified error(s) occurred while reading the item. No item ID was returned.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No item ID was returned.
MP_E_INV_SESSION	An invalid session handle was

	used for *SPlusSession* parameter. No item ID was returned.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. No item ID was returned.
SP_E_USER	The recipient descriptor was incomplete or specified incorrectly. No item ID was returned.
SP_E_NO_SCHEDULE	No schedule was located for the user. No item ID was returned.
SP_E_RESTRICTION	The restriction was specified incorrectly. No data was returned.
SP_E_ITEM	The item ID was invalid. The item ID may have been deleted by another process. No item ID was returned.
SP_E_TIME	The *Restriction* parameter was not specified properly. The format expected was YYY/MM/DD.
SP_E_ADDRESS_FORMAT	The recipient descriptor was incomplete or specified incorrectly. No information was returned.
SP_E_NO_ITEMS	There were no remaining items that matched the restriction.
SP_E_NOT_INSTALLED	Schedule+ was not installed.
SP_E_NO_PRIVILEGE	The caller did not have a privilege level of **Read** or greater for the user's schedule.
SUCCESS_SUCCESS	The function returned successfully.

See Also

SPBegin

SPDelete
SPEnd
SPReadAppt
SPReadTask

SPReadAppt

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPReadAppt(
    SPlusSession As Numeric
    UIParam As Numeric
    SPlusUser As MapiRecip
    ItemID As String
    Flags As Numeric
    Reserved As Numeric
    Appt As SPlusAppt
    Assoc() As SPlusAssoc
    Attendees() As SPlusAttd
    UserType As String
 ) as Numeric
```

Description

This function reads an appointment from a user's schedule when provided with the appointment ID.

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd.** If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. Otherwise, calls with *SPlusSession* equal to 0 return MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
SPlusUser	A recipient descriptor that specifies the user. The *Name* and *Address* fields must be completed. An empty string specifies the currently signed-in user.
ItemID	The string identifier of the item.
Flags	A bitmask of flags. All unused flags are reserved and

should be set to 0. The following flag is defined in MAPILIB.PRG:

```
#define SP_SUPPRESS_RECIPIENTS   4096
```

When this flag is present, no attendees are read into the appointment type and the *AttendCnt* field of **SPlusAppt** is set to 0 even if attendees have been invited to the appointment.

Reserved	Reserved for future use. This parameter must be 0.
Appt	An appointment type containing the item contents.
Assoc ( )	An array of associated item descriptors. In this release, only alarms are allowed.
Attendees ( )	A dynamic array of attendee descriptors specifying the meeting's attendees and their confirmation status.
UserType	A string specifying the user type. For this release, the two supported types are **Individual** and **Unspecified Resource**.

Return Value	**Meaning**
MP_E_DISK_FULL	The disk was full. The item was not read.
MP_E_FAILURE	Unspecified error(s) occurred while reading the message. The item was not read.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. The item was not read.
MP_E_INV_SESSION	An invalid session handle was used for the *SPlusSession* parameter. The item was not read.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. The item was not read.
SP_E_USER	The recipient descriptor was incomplete or specified incorrectly. The item was not read.
SP_E_NO_SCHEDULE	No schedule was located for the user. The item was not read.

SP_E_NO_PRIVILEGE	The caller did not have a privilege level of **Read** or greater for the user’s schedule. The item was not read.
SP_E_ITEM	The item ID was invalid. The item ID may have been deleted by another process. The ID may have been the wrong type (for example, an ID for a task). The item was not read.
SUCCESS_SUCCESS	The function returned successfully.

See Also

SPBegin

SPEnd

SPlusAppt

SPFreeBusy

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPFreeBusy(
    SPlusSession As Numeric
    UIParam As Numeric
    UserCount As Numeric
    Users() as MapiRecip
    Month As String
    Flags As Numeric
    Reserved As Numeric
    HaveInfo As Reference to a String
    FreeBusy() As String
 ) as Numeric
```

Description

This function determines commonly held free time periods (in half-hour intervals) for a group of users. Free or busy information for each of the users is fetched and then merged to determine periods when none of the specified users is busy.

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd**. If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. Otherwise, calls with *SPlusSession* equal to 0 return MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
UserCount	The number of users information is being retrieved for. A value of 0 specifies the retrieval of free/busy information for the currently signed-in user.
Users ( )	An array of *UserCount* recipient descriptors specifying users. A descriptor with all fields 0 or empty strings (as appropriate) indicates the currently signed-in user. Otherwise, the *Name* and *Address* fields must be completed. This parameter must be NULL if *UserCount* is 0.
Month	A string specifying the month that information is retrieved for. The format is YYYY/M for months 1 through 9 or YYYY/MM for months 10 through 12. The year must be in the range 1920–2019.
Flags	A bitmask of flags. All flags are reserved and should be set to 0.
Reserved	Reserved for future use. This parameter must be 0.
HaveInfo	A string of length *UserCount* characters. After a successful call to this function, the first character is set to **Y** (yes) or **N** (no) to indicate whether free/busy information for the first user was retrieved. Each subsequent character is set in the same way for each remaining user.
FreeBusy ( )	An array of 31 strings. Each string contains 48 characters. Each successful call to this function fills the strings with free/busy information for each day of the month. The first string contains data for the first day of the month, the second string contains data for the second day of the month, and so on. In each string, the function inserts **F** (free) in the first character of the string if none of the specified users are busy from midnight to 12:30 A.M., and **B** (busy) otherwise. The second character in the string is set to **F** or **B** for the period from 12:30 A.M. to 1:00 A.M., and

so on, working around the clock in half-hour intervals.

Return Value	**Meaning**
MP_E_FAILURE	Unspecified error(s) occurred while performing this operation. No data was returned.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No data was returned.
MP_E_INV_SESSION	An invalid session handle was used for the *SPlusSession* parameter. No data was returned.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No data was returned.
MP_USER_ABORT	The user canceled the sign-in. No data was returned.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. No data was returned.
SP_E_USER	A recipient descriptor specified in the *Users* array was incomplete. No data was returned.
SP_E_MONTH	The month was specified incorrectly. No data was returned.
SP_E_NOT_INSTALLED	Schedule+ was not installed.
SUCCESS_SUCCESS	The function returned successfully.

See Also

SPBegin

SPEnd

SPReadTask

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPReadTask(
    SPlusSession As Numeric
    UIParam As Numeric
    SPlusUser As MapiRecip
    ItemID As String
    Flags As Numeric
    Reserved As Numeric
    Task As SPlusTask
    Assoc() As SPlusAssoc
    Attendees() As SPlusAttd
    UserType As String
 ) as Numeric
```

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd.** If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. Otherwise, calls with *SPlusSession* equal to 0 return MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
SPlusUser	A recipient descriptor specifying the user. The *Name* and *Address* fields must be completed. A descriptor with all fields equal to 0 or empty strings (as appropriate) specifies the currently signed-in user.
ItemID	The string identifier of the item.
Flags	A bitmask of flags. All flags are reserved and should be set to 0.
Reserved	Reserved for future use. This parameter must be 0.
Task	A **SPlusTask** type containing the item contents.
Assoc ( )	An array of associated item descriptors. In this release, only alarms are allowed.
Attendees ( )	Reserved for future use.
UserType	A string specifying the user type. For this release, the

two supported types are **Individual** and **Unspecified Resource**.

Return Value	Meaning
MP_E_DISK_FULL	The disk was full. The item was not read.
MP_E_FAILURE	Unspecified error(s) occurred while reading the message. The item was not read.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. The item was not read.
MP_E_INV_SESSION	An invalid session handle was used for the *SPlusSession* parameter. No item was read.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. The item was not read.
SP_E_USER	The recipient descriptor was incomplete or specified incorrectly. The item was not read.
SP_E_NO_SCHEDULE	No schedule was located for the user. The item was not read.
SP_E_NO_PRIVILEGE	The caller didn't have a privilege level of **Read** or greater for the user's schedule.
SP_E_ITEM	The item ID was invalid. The item ID may have been deleted by another process. The ID may have been the wrong type (for example, an ID for an appointment). The item was not read.
SUCCESS_SUCCESS	The function returned successfully.

See Also

SPBegin

SPEnd

SPlusTask

SPUserInfo

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPUserInfo(
    SPlusSession As Numeric
    UIParam As Numeric
    SPlusUser As MapiRecip
    Flags As Numeric
    Reserved As Numeric
    UserInfo As SPlusUser
 ) as Numeric
```

Description

This function determines the identity of the user's assistant, whether the user is a resource, whether copies of meeting requests should go to the user as well as the user's assistant. The working hours for the logged in user are also returned.

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd.** If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. Otherwise, calls with *SPlusSession* equal to 0 return MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
SPlusUser	A recipient descriptor specifying the user. The *Name* and *Address* fields must be completed. A descriptor with all fields equal to 0 or empty strings (as appropriate) specifies the currently signed-in user.
Flags	A bitmask of flags. All flags are reserved and should be set to 0.
Reserved	Reserved for future use. This parameter must be 0.
UserInfo	An **SPUserInfo** type containing information about the user.

Return Value	Meaning
MP_E_FAILURE	Unspecified error(s) occurred while retrieving the information. No information was returned.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed. No information was returned.
MP_E_INV_SESSION	An invalid session handle was used for the *SPlusSession* parameter. No information was returned.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No information was returned.
MP_USER_ABORT	The user canceled the process. No information was returned.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. No information was returned.
SP_E_ADDRESS_FORMAT	The recipient descriptor was incomplete or specified incorrectly. No information was returned.
SP_E_NO_SCHEDULE	No schedule information was located for the user. Default information for the user was returned.
SP_E_NOT_INSTALLED	Schedule+ was not installed.
SP_E_NOT_ONLINE	The user was offline when this function was called.
SUCCESS_SUCCESS	The function returned successfully.

See Also

SPBegin

SPEnd

SPSaveAppt

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPSaveAppt(
    SPlusSession As Numeric
    UIParam As Numeric
    SPlusUser As MapiRecip
    Appt As SPlusAppt
    Assoc() As SPlusAssoc
    Attendees() As SPlusAttd
    UserType As String
    Flags As Numeric
    Reserved As Numeric
    ItemID As Reference to a String
 ) as Numeric
```

Description

If an appointment with the string *ItemID* exists, SPSaveAppt will modify the elements of *Appt struct* according to *Flags.*

If the string *ItemID* is empty, a new appointment is created with the fields specified by the *Appt* parameter. The new appointment's ID is returned in the *ItemID* parameter on completion of the call

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd.** If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. Otherwise, calls with *SPlusSession* equal to 0 return MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
SPlusUser	A recipient descriptor that specifies the user. The *Name* and *Address* fields must be completed. A descriptor with all fields equal to 0 or empty strings (as appropriate) specifies the currently signed-in user.
Appt	A **SPlusAppt** type. *ItemType* This field must be **SimpleAppt**, **OrganizedMeeting**,

	or **BookedMeeting**. The item type of an existing appointment cannot be modified.
Assoc ( )	An array of associated item descriptors. In this release, only alarms are allowed. If an alarm is specified for the appointment, its ring time must be later than the current time. Otherwise this function returns SP_E_ALARM_RING_IN_PAST.
Attendees ( )	An array of attendee descriptors specifying the meeting's attendees and their confirmation status.
UserType	A string specifying the user type. For this release, the two supported types are **Individual** and **Unspecified Resource**.

Flags	A bitmask of flags. You should always set unspecified flags to 0. Undocumented flags are reserved. This function supports the same flags as **MPSaveMail**.
	The following additional flags defined in SPLUS.BAS or WRKGROUP.MDB are also supported:
	`#define SP_DEFAULT_ALARM   4096`
	`#define SP_MOD_FLAGS   65536`
	`#define SP_MOD_ASSOC   131072`
	`#define SP_MOD_ATTENDEES   262144`
	`#define SP_MOD_TEXT   524288`
	`#define SP_MOD_TIMES   4194304`
	SP_DEFAULT_ALARM sets the default alarm for an appointment to the user's Schedule+ preference. The flag overrides any settings in the associated items array. If the alarm on an existing appointment is being modified, this flag removes the old alarm and sets it to the default.
	SP_MOD_FLAGS, SP_MOD_ASSOC_ITEMS, SP_MOD_ATTENDEES, SP_MOD_TEXT, and SP_MOD_TIMES may be used only when modifying an existing appointment. In this case, the set flags indicate which fields should be modified. Note that some fields of the appointment in **SPlusAppt** cannot be modified (*ItemType, Creator, Organizer,* and *OrgItemID).*

Reserved	Reserved for future use. This parameter must be 0.
ItemID	A string identifier for this item. An empty string specifies a new appointment. Item ID strings must be allocated by the caller and must be able to hold at least 64 characters.

Return Value	**Meaning**
MP_E_DISK_FULL	The disk was full.
MP_E_FAILURE	Unspecified error(s) occurred while saving the item.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed.
MP_E_INV_SESSION	An invalid session handle was used for the *SPlusSession* parameter. No changes were saved.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No changes were saved.
MP_USER_ABORT	The user canceled the process. No changes were saved.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. No changes were saved.
SP_E_NOT_INSTALLED	This function could not be performed because Schedule+ was not installed.
SP_E_USER	The recipient descriptor was incomplete or specified incorrectly. No changes were saved.
SP_E_ADDRESS_FORMAT	The recipient descriptor was incomplete or specified incorrectly. No information was returned.
SP_E_NO_SCHEDULE	No schedule was located for the user. No changes were saved.
SP_E_NO_PRIVILEGE	The caller did not have privilege level of **Create** or greater for the user's schedule.
SP_E_ITEM	The item ID was invalid. The item ID may have been

	deleted by another process. No changes were saved.
SP_E_TYPE	An invalid item type was supplied in the **SPlusAppt** type.
SP_E_FLAGS	An invalid flag was supplied in the **SPlusAppt** type.
SP_E_ASSOC	There was an error in the way the associated items were specified.
SP_E_ALARM_RING_IN_PAST	An alarm was specified whose ring time has passed. No changes were saved.
SP_E_ORGANIZER	The *Organizer* field of the **SPlusAppt** type was not specified properly.
SP_E_ORG_ID	The *OrgItemID* field of the **SPlusAppt** type was not specified properly.
SP_E_ATTENDEES	The attendees were not specified correctly. No changes were saved.
SP_E_TIME	The *DateStart* or *DateEnd* field of the **SPlusAppt** type was not specified properly.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPSaveMail

SPBegin

SPEnd

SPlusAppt

SPSaveTask

{bmc retrnval.bmp}{bmc seealso.bmp}

```
SPSaveTask(
    SPlusSession As Numeric
    UIParam As Numeric
    SPlusUser As MapiRecip
    Task As SPlusTask
    Assoc() As SPlusAssoc
    Attendees() As SPlusAttd
    UserType As String
    Flags As Numeric
    Reserved As Numeric
    ItemID As Reference to a String
 ) as Numeric
```

Description

If a task with the *ItemID* exists, the fields of the task are replaced by the elements of the *Task* parameter.

If the string *ItemID* is empty, a new task is created with the fields specified by the *Task* parameter. The new task's ID is returned in the *ItemID* parameter on completion of the call.

Parameter	Description
SPlusSession	An opaque session handle whose value represents a session with the scheduling system. Session handles are returned by **SPBegin** and invalidated by **SPEnd.** If the value is 0, a session is established from a system default session (if one exists) or with a sign-in dialog box. Otherwise, calls with *SPlusSession* equal to 0 return MP_E_INV_SESSION.
UIParam	The parent window handle for the dialog box (of type HWND) cast to a NUMERIC.
SPlusUser	A recipient descriptor specifying the user. The *Name* and *Address* fields must be completed. A descriptor with all fields 0 or empty strings (as appropriate) specifies the currently signed-in user.
Task	An **SPlusTask** type containing the task contents. *ItemType* This field must be **SimpleTask**. The item type of an

	existing task cannot be modified.
Assoc ( )	An array of associated item descriptors. In this release, only alarms are allowed. If an alarm is specified for the task, its ring time must be later than the current time. Otherwise this function returns SP_E_ALARM_RING_IN_PAST.
Attendees ( )	Reserved for future use.
UserType	A string specifying the user type. For this release, the two supported types are **Individual** and **Unspecified Resource**.
Flags	A bitmask of flags. You should always set unspecified flags to 0. Undocumented flags are reserved. This function supports the same flags as **MPSaveMail**. The following additional flags defined in MAPILIB.PRG are also supported: #define SP_DEFAULT_ALARM 4096 #define SP_MOD_FLAGS 65536 #define SP_MOD_ASSOC 131072 #define SP_MOD_TEXT 524288 #define SP_MOD_TIMES 4194304 #define SP_MOD_PROJECT_NAME 8388608 #define SP_MOD_PRIORITY 16777216 SP_DEFAULT_ALARM specifies the user's default alarm setting. This flag overrides any setting in the associated items array of **SPlusAssoc**. If the alarm on an existing task is being modified, this flag removes the old alarm and sets it to the default. SP_MOD_FLAGS, SP_MOD_ASSOC_ITEMS, SP_MOD_TEXT, SP_MOD_TIMES, SP_MOD_PROJECT, and SP_MOD_PRIORITY may be used only when modifying an existing task. In this case, the set flags indicate which fields should be modified. Note that some fields of the task cannot be modified (for example, the *ItemType* and *Creator* fields of **SPlusTask**).
Reserved	Reserved for future use. This parameter must be 0.
ItemID	A string identifier for this item. An empty string specifies a new task. Item ID strings must be allocated by the caller and must be able to hold at least 64 characters.

Return Value	**Meaning**
MP_E_DISK_FULL	The disk was full.

MP_E_FAILURE	Unspecified error(s) occurred while saving the item.
MP_E_INSUFFICIENT_MEMORY	There was insufficient memory to proceed.
MP_E_INV_SESSION	An invalid session handle was used for the *SPlusSession* parameter. No changes were saved.
MP_E_LOGIN_FAILURE	There was no default sign-in, and the user failed to sign in successfully when the sign-in dialog box was displayed. No changes were saved.
MP_USER_ABORT	The user canceled the process. No changes were saved.
MP_E_NOT_SUPPORTED	The *Flags* or *Reserved* parameter was not set correctly. No changes were saved.
SP_E_NOT_INSTALLED	This function could not be performed because Schedule+ was not installed.
SP_E_USER	The recipient descriptor was incomplete or specified incorrectly. No changes were saved.
SP_E_NO_SCHEDULE	No schedule was located for the user. No changes were saved.
SP_E_NO_PRIVILEGE	The caller did not have privilege level of **Create** or greater for the user's schedule.
SP_E_ITEM	The item ID was invalid. The item ID may have been deleted by another process. No changes were saved.
SP_E_TYPE	An invalid item type was supplied in the **SPlusTask** type.
SP_E_FLAGS	An invalid flag was supplied in

	the **SPlusTask** type.
SP_E_ASSOC	There was an error in the way the associated items were specified.
SP_E_ALARM_RING_IN_PAST	An alarm was specified whose ring time is in the past. No changes were saved.
SP_E_TIME	The *DateStart* or *DateEnd* field in **SPlusTask** was not specified properly.
SP_E_PRIORITY	The *Priority* field of the **SPlusTask** type was specified incorrectly.
SUCCESS_SUCCESS	The function returned successfully.

See Also

MPSaveMail

SPBegin

SPEnd

SPlusAssoc

SPlusTask

Return Values for MAPI/SPLUS Functions

```
#define SUCCESS_SUCCESS                 0
#define MP_USER_ABORT               1
#define MP_E_FAILURE                2
#define MP_E_LOGIN_FAILURE          3
#define MP_E_DISK_FULL              4
#define MP_E_INSUFFICIENT_MEMORY    5
#define MP_E_ACCESS_DENIED          6
#define MP_E_TM_SESSIONS            8
#define MP_E_TM_FILES               9
#define MP_E_TM_RECIPIENTS          10
#define MP_E_ATT_NOT_FOUND          11
#define MP_E_ATT_OPEN_FAILURE       12
```

```
#define MP_E_ATT_WRITE_FAILURE          13
#define MP_E_UNKNOWN_RECIPIENT          14
#define MP_E_BAD_RECIPTYPE              15
#define MP_E_NO_MESSAGES                16
#define MP_E_INV_MESSAGE                17
#define MP_E_TEXT_TOO_LARGE             18
#define MP_E_INV_SESSION                        19
#define MP_E_TYPE_NOT_SUPPORTED         20
#define MP_E_AMB_RECIPIENT              21
#define MP_E_MSG_IN_USE                 22
#define MP_E_NETWORK_FAILURE            23
#define MP_E_INV_EDITFIELDS             24
#define MP_E_INV_RECIPS                 25
#define MP_E_NOT_SUPPORTED              26
#define SP_NO_REQUEST_SENT              0
#define SP_NO_RESPONSE                  1
#define SP_POSITIVE_RESPONSE            2
#define SP_NEGATIVE_RESPONSE            3
#define SP_AMBIGUOUS_RESPONSE           4
#define SP_RESPONSE_REQUESTED           65536
#define SP_PRIVATE                      1
#define SP_TENTATIVE                    4
#define SP_PRV_PROJECT                  2
#define SP_BOSS_WANTS_COPY              1
#define SP_DEFAULT_ALARM                4096
#define SP_MOD_FLAGS                    65536
#define SP_MOD_ASSOC                    131072
#define SP_MOD_ATTENDEES                262144
#define SP_MOD_TEXT                     524288
#define SP_MOD_TIMES                    4194304
#define SP_MOD_PROJECT_NAME             8388608
#define SP_MOD_PRIORITY                 16777216
#define SP_MOD_ALL                              33488896
#define SP_SUPPRESS_RECIPIENTS          4096
```

```
#define SP_E_INV_SENT_FOR              10000
#define SP_E_NOT_INSTALLED             10005
#define SP_E_NO_ITEMS                  10010
#define SP_E_NO_SCHEDULE               10015
#define SP_E_NO_PRIVILEGE              10020
#define SP_E_ADDRESS_FORMAT            10025
#define SP_E_USER                      10030
#define SP_E_ITEM                      10035
#define SP_E_TYPE                      10040
#define SP_E_FLAGS                              10045
#define SP_E_ASSOC                              10050
#define SP_E_ORGANIZER                 10055
#define SP_E_ORG_ID                    10060
#define SP_E_ATTENDEES                 10065
#define SP_E_TIME                      10070
#define SP_E_PRIORITY                  10075
#define SP_E_MONTH                     10080
#define SP_E_RESTRICTION                        10085
#define SP_E_ALARM_RING_IN_PAST        10090
#define SP_E_NOT_ONLINE                10095
#define SUCCESS_NO_SCHEDULE            10100
```

Help Instructions

The following information is here to help you quickly find what you need in Help. For more information about Help, choose How to Use Help from the Help menu.

{bmr hand.bmp}	The pointer becomes a hand when it is over a region that contains more information (a pop-up topic) or a jump to another topic. Click the mouse in this region to jump to the topic or to view the pop-up topic.
{bmr ulgrntxt.bmp}	To jump to another topic, choose this highlighted text.
{bmr dugrntxt.bmp}	To display a pop-up topic containing art, a definition, a note, or further information about this topic, choose this highlighted text. To clear a pop-up topic, click anywhere or press ESCAPE.

{bmr back.bmp}	To return to the last screen of information, click the Back button, or press B.
{bmr retrnval.bmp}	To see a list of the function's return values, choose the Return Values button. This button is present only when the current topic has a Return Values section.
{bmr seealso.bmp}	To see a list of topics related to the current topic, choose the See Also button. This button is present only when the current topic has a See Also section.
{bmr example.bmp}	To display an example appropriate to the topic, choose the Example button. This button is present only when the current topic has an example.

Microsoft Support Services

FOXMAPI.FLL is a "layer" on top of MAPI.DLL and SPLUS.DLL. This layer provides cursors to replace the C-style structures that the actual DLLs expose. Apart from this mapping of C data types to FoxPro data types, there is almost no extra code in FOXMAPI.FLL. Thus when calls to FOXMAPI behave unexpectedly it is likely a problem with calling MAPI.DLL (or SPLUS.DLL). Because of this, support for FOXMAPI.FLL is provided as part of general MAPI and SPLUS support.

More information on the MAPI and SPLUS interfaces can be found on CompuServe on the MSWRKGR forum. Section 15 is entitled MS Schedule+, Section 16 is entitled MS Eforms, and Section 17 is entitled MAPI.

Note that detailed support will NOT be available in foxforum. While they may be able to answer basic questions, more complex MAPI and SPLUS functionality questions should be directed first to the MSWRKGRP section. Note that because FOXMAPI.FLL uses the same function calls, structures, flags, etc. as the original MAPI and SPLUS dynamic link libraries,   you should read the documentation for those DLLs and work out how to write the corresponding code in FoxPro.

Microsoft TDD/TT (Text Telephone) Support

Microsoft Product Support Services is available for the deaf and hard-of-hearing. Using a special TDD/TT modem, dial (206) 635-4948. Call between 6:00 A.M. and 6:00 P.M. Pacific time, Monday through Friday.

CompuServe

The Microsoft Connection on CompuServe provides online technical information for Microsoft products. With The Microsoft Connection, you can exchange messages with Microsoft support engineers and experienced Microsoft users, and you can download software such as drivers, patches, tools, and add-ons, provided by Microsoft and CompuServe members at no charge.

By using The Microsoft Connection, you can access the Microsoft Developer Services area. You are encouraged to use this area to speak directly to Microsoft about developer-related issues. The Microsoft Developer Services area offers the following advantages:

{bmc b.bmp} Developer Forums. These forums cover information about Windows, languages, tools, and utilities from a developer's perspective. For example, the Windows SDK Forum provides information about programming for Windows. The section leads for these forums are from Microsoft Product Support and can help answer your questions about the Windows application programming interface (API).

{bmc b.bmp} Confidential Technical Service Requests. Microsoft offers private (fee-based per incident) technical support to help solve your more complex development problems. For more details, see the Microsoft Developer Services area.

{bmc b.bmp} Developer Knowledge Base. This up-to-date reference tool, compiled by Microsoft Product Support, contains developer-specific technical information about Microsoft products.

{bmc b.bmp} Software Library. You can search this collection of text and graphics files, sample code, and utilities by keyword, and download the files for local use.

To connect to The Microsoft Connection, type GO MICROSOFT at the CompuServe "!" prompt. For information about establishing a CompuServe account, call (800) 848-8199, 8:00 A.M. to 10:00 P.M. eastern standard time. Ask for operator 230 and receive a $15 connect-time usage credit.