MAPI Component for Borland Delphi 1.0

Release Notes

Version 1.0

This version of the MAPI component for Delphi was developed and has been tested under Windows 95 test release build 950, the mail client, MSN and the fax service using LAN and RAS connections.

Introduction

The MAPI component is an easy-to-use, yet powerful piece of software that capsulates the set of unhandy SIMPLE MAPI API functions into a friendly set of Delphi-style methods, events and properties.

The MAPI component allows you to easily mail-enable your applications without requiring any VBX or OCX. It supports up to 255 recipients as well as 255 CC's (carbon copies) and 255 BCC's (blind carbon copies). You may attach up to 255 files as well.

A main target for development was to expose a fault tolerant interface. This means, for example, that you can call any method at any time; if a connection could be established, the component intelligently decides which steps have to be done to complete the requested task. If you call the send method, for example, the component first checks, if a session has been established and opens one, if necessary. It then looks into the recipient list to find out if you supplied any recipients. If so, it validates all entries and calls common dialogs to resolve unknown names; otherwise, it opens the address book to let the user select recipients. If you didn't supply a subject, the component opens the common dialog for creating a message.

Installing the MAPI component

Copy the files MAPI.DCR and MAPI.DCU into your Delphi\lib directory. Then add the MAPI component to your library from Delphi with Options\Install components\Add. The component will install into a new folder on your VCL toolbar called "C/S Pro".

Using the MAPI component

Visual Properties

Active (Boolean)

Indicates if the MAPI component has an active MAPI link or not. You may set the state of the MAPI component through this property instead of using the Open and Close methods.

AutoDownload (Boolean)

When this property is True, the component downloads new mail when establishing a mail session using the method Open.

RecipientList (TStringList)

This dynamic string array contains the names of all recipients of a retrieved or prepared message. You may fill in names code-driven into this property or call the AddressBook method to define recipients. If you supply the recipients by code the component verifies each entry before sending the message.

CCList (TStringList)

Same as recipient list for carbon copy recipients.

BCCList (TStringList)

Same as recipient list for blind carbon copy recipients.

KeepConnection (Boolean)

If this property is true, the component holds a connection until you close it or the component is destroyed. This might be handy if your application wants to send more than one message. If false, the component shuts down the active MAPI session after a message has been sent.

MarkRead (Boolean)

This property controls if fetched messages should be marked as read or not. **Important: To mark messages as read you must have actually "read" the message using GetMessage(<MessageID>). FirstMessage/NextMessage do not set the read flag.**

NewMessagesOnly (Boolean)

This property tells the component, if calls to FirstMessage or NextMessage should return all messages in the inbox or only new (unread) messages.

WizardMode (Boolean)

If this property is true, the component intelligently opens common dialogs to complete the information necessary to send a message. If false, the component raises errors if the information for the requested task is incomplete.

ProfileName (String)

If you want automated logon to MAPI without being asked for a logon ID, you should supply a profile name (if you are using the Windows 95 Exchange client, it's most likely "MS Exchange Settings").

Password (String)

If you must supply a password for automated logon, enter it here.

Subject (String)

Subject (header) of the message. This property is used when sending and reading mail. If this property is blank when you call the Send method and wizard mode is active , a common "new mail"-dialog will be presented, because a mail without subject is bad style.

MessageText (TStringList)

This property holds the message text (body) of a message, that has been read or is to be sent. **Please note that for performance reasons the methods FirstMessage and NextMessage do not retrieve the message body. You must call the GetMessage method in order to have access to the message body.**

FileList (TStringList)

This list holds the full path names of all attached files of a retrieved or prepared message. **Please note that for performance reasons the methods FirstMessage and NextMessage do not retrieve the list of attached files. You must call the GetMessage method in order to have access to the attached files.**

Interval (Integer)

Allows you to activate the internal polling mechanism. Set this property to the polling interval (in minutes) you like (up to 65'000). Everytime the components detects new mail in your inbox you'll get an OnNewMessages event. You must, however, (re)open your MAPI component in order to get access to those new mails.

Runtime Properties

MessageID (String)

This property holds a unique identifier which you need to fully retrieve a message using the method GetMessage(MessageID). If you loop through all messages with FirstMessage / NextMessage, you should buffer the MessageID's in an array to be able to retrieve a specific message body later.

MessageClass (String)

This property sets a message class filter for sending and receiving custom messages. If you wish to process normal (interpersonal) mail, leave this property blank.

Originator (String)

This property contains the originator's name after you fetched a specific message using NextMessage. It has no effect when sending mails.

DateReceived (TDateTime)

This property contains the date and time of a received message when a specific message has been fetched by FirstMessage or NextMessage. It is ignored when sending a message.

hasFiles (Boolean)

Indicates if a fetched message contains attached files. This property is ignored when sending messages.

EOF

This property indicates that the last message has been fetched using the method NextMessage and allows you to control a fetch loop like this:

```
FirstMessage;
while not MAPI1.EOF do begin
  <store MessageID and assign msg properties to visual controls or variables>
  NextMessage;
end;
```

Please note that the call to NextMessage that sets EOF to true does not deliver a valid message anymore.

Methods

Open

Logs you on to the mail postoffice. It is not necessary to call this method because the component calls it by itself when necessary. If the user cancels the logon, an exception of type EMAPIError is raised.

Close

Shuts down an active mail session. You do not have to call this method explicitly because the component calls it when your application exits.

ResolveNames

This method tries to resolve all names in the RecipientList, the CCList and the BCCList. If WizardMode is True, then a common dialog is presented to maunally resolve unknown recipients. If WizardMode is false, unresolvable names are removed from the list. This method is called automatically before a message is sent. You can use the method, however, to let the user visually check recipient entries before sending a message.
If there are unresolvable names an exception of type EMAPIError is raised.

AddressBook

Opens the common address book dialog.

Send

Sends a prepared mail. If WizardMode is active, you do not have to supply all necessary information for sending a message. The component will ask you for the missing information automatically. Otherwise, exceptions are generated for each missing item.

Clear

Clears all message properties. You typically call this method before preparing a new message.

FirstMessage

Reads the header information of the first message. The property NewMessagesOnly decides if all or only new (unread) messages are fetched. **Please note that this function is relatively slow. This is not due to poor implementation (this method calls MAPIFindNext straightforward); in fact, mail clients like MS Mail buffer retrieved messages locally and fetch only new messages. It is recommended to use it the same way, if you intend to write your own mail client.**

NextMessage

Reads the header information of the next message. The property NewMessagesOnly decides, if all or only new (unread) messages are fetched. You may use the methods FirstMessage and NextMessage the following way:

```
FirstMessage;
  while not MAPI1.EOF do begin
    <assing msg properties to visual controls or variables>
    NextMessage;
  end;
```

Please note that the call to NextMessage that sets EOF to true does not deliver a valid message anymore and that this function is relatively slow. This is not due to poor implementation (this method calls MAPIFindNext straightforward); in fact, mail clients like MS mail buffer retrieved messages locally and fetch only new messages. It is recommended to use it the same way, if you intend to write your own mail client.

GetMessage(MessageID: String)

This message gets the full information of a previously fetched message. After this call, all properties, including Text and FileList (if hasFiles is True) are available. You must pass the MessageID of a previously fetched message (i.e.using FirstMessage or NextMessage) to this method in order to identify a specific message. If you fill all message headers into a list box, for example, a call to this method could look like this:

DeleteMessage(MessageID: String)

This message deletes a message permanently from the message store. You must pass the MessageID of a previously fetched message (i.e.using FirstMessage or NextMessage) to this method in order to identify a specific message.

Events

OnNewMessages(Sender: Tobject)

Occurs when the polling mechanism is active (i.e. Interval > 0) and new mail has been detected. To process those messages you must Open (or Close/Open) your MAPI component and read those messages using FirstMessage / NextMessage. A read loop in this event could look like this:

```
MAPI1.Close; { Just for sure }
MAPI1.Open;
MAPI1.FirstMessage;
while not MAPI1.EOF do begin
  MAPI1.GetMessage(MAPI1.MessageID);
  { Process the message here }
  MAPI1.NextMessage;
end;
```

OnStateChange(Sender: Tobject)

This event notifies you when the Active property of the MAPI component has changed. This can be useful to visually display if the application has an active MAPI link or not.