PermitUserLogin property

```
property PermitUserLogin: Boolean; { TVimSession }
```

If set to TRUE and either the LoginName or Password properties are empty then a dialog will pop up asking for the users details.

ccMail Custom Controls

The ccMail Custom Controls (TVimSession ,TVimSendMessage, TVimInbox, and TVimAddressBook) are a collection of Delphi VCL files for sending and receiving messages using the VIM transport mechanism..

TVimAddressBook

This component provide access to the information help in the ccMail / Notes address book. An example of how to use this object is contained in the DEMO3PRJ demonstration program.

See also :-

properties	**methods**
Active	
ActiveAddressBook	
Session	

run-time properties

AddressBookList

Gateways

Names

data structures and support classes

TVimAddress

TVimSession

The TVimSession class provides a channel to the ccMail engine as defined in the VIM (Vendor Independent Messaging) API. It is used by the other components to provide access to various VIM functions..

See also:

properties	**methods**
Active	
LoginName	
Password	
PermitUserLogin	
PostOfficePath	
run-time only properties	
VimError	
ErrorMessage	
ExtendedError	
ApplicationError	

TVimSendMessage

Used to create and send messages via ccMail.

See also:

Properties	**Methods**
Attachments	Send
BCC	
CC	
MessageTxt	
MessageLines	
Priority	
Receipt	
Recipients	
Session	
Subject	

Active property

```
property Active : boolean; { TVimAddressBook }
```

Setting this to true causes the AddressBookList to be populated with the names of the available address books. The component must be active before the details of individual addresses may be accessed.

For ccMail the default address book is 'Directory' - additional address books are usually 'Public Mailing Lists' and 'Private Mailing Lists'.

Active property

```
property Active: Boolean; { TVimSession }
```

The Active property is used to query or set the state of TVimSession.

Setting the property to True forces a login to ccMail to be attempted. If the login fails then the VimError property will be set to a non-zero value and Active will be set to false.

Conversely setting the property to False logs out the current ccMail user.

See also:

VimError property

ErrorMessage property

```
property ErrorMessage: String; { TVimSession }
```

Contains the error message returned by the VIM engine whenever an error is encountered by either of TVimSession and TVimSendMessage .

See also

VimError

ExtendedError

ApplicationError

VimError property

```
property VimError: VimStatus; { TVimSession }
```

The VimError property is read-only and contains a non-zero value if the last ccMail activity failed.

See also

ErrorMessage

ExtendedError

ApplicationError

Password property

```
property Password: String; { TVimSession }
```

Must be a the valid ccMail password for the entity refered to by LoginName. property

Attachments property

```
property Attachments: TStrings { TVimSendMessage }
```

List of fully qualified file names for attachment to the message. If a file is not found the Send method will fail.

PostOfficePath property

```
property PostOfficePath: String; { TVimSession }
```

This property contains the fully qualified path of the ccMail post office to be used for this session. It is set up automatically when the Active property is set to true if no value has been entered.

LoginName property

```
property LoginName: String; { TVimSession }
```

The ccMail user from whom the message will be sent.

TVimInboxMsg

The following Delphi definition describes the class used to store additional details of a ccMail / Notes message within a TVimInbox object. It is stored in the InboxContents TStrings property as the associated object.

```
type
  TVimInboxMsg = class(TObject)
  public
    property Date : TDateTime;
    property Priority : TVimPriority;
    property MsgUnRead : boolean;
    property From:string;
  end;
```

ExtendedError Property

```
property ExtendedError: String; { TVimSession }
```

Contains any additional error text returned by the VIM engine after a TVimSession or TVimSendMessage method has failed..

See Also

ErrorMessage

ApplicationError

VimError

Licence Agreement

ApplicationError Property

```
property ApplicationError: String; { TVimSession }
```

This will contain additional information specific to the TVimSession or TVimSendMessage method that has failed.

List of files

The following is a list of files supplied as part of this package

VIMSESS.DCU	TVimSession
VIMSEND.DCU	TVimSendMessage
VIMINBOX.DCU	TVimInbox
VIMADDBK.DCU	TVimAddressBook
VIM.DCU	Support routines
VLOGIN.DCU	Required for the ccMail login/password form
VLOGIN.DFM	The Login/Password form
VLOGIN.PAS	The source for above
	Do not replace the supplied VLOGIN.DCU, only VLOGIN.DFM
VABOUT.DCU	About form
VABOUT.DFM	
VIMVCL.HLP	This file
VIMVCL.KWF	Keyword file to link VIMVCL.HLP into the Delphi IDE
DEMO1PRJ.DPR	Sample project for TVimSendMessage
DEMO1SRC.PAS	
DEMO1SRC.DFM	
DEMO2PRJ.DPR	Sample project for TVimInbox
DEMO2SRC.PAS	
DEMO2SRC.DFM	
DEMO3PRJ.DPR	Sample project for TVimAddressBook
DEMO3SRC.PAS	
DEMO3SRC.DFM	

Installation

Copyright

Send Method

```
function Send : Boolean; { TVimSendMessage }
```

Initiates the creation and sending of the message via ccMail. Returns false if the message could not be sent.

See also

ErrorMessage

ExtendedError

ApplicationError

VimError

BCC Property

```
property BCC: TStrings; { TVimSendMessage }
```

List of ccMail addresses who will be Blind Carbon Copied on the message. If any of the supplied addresses are invalid the Send method will fail.

CC Property

```
property CC: TStrings; { TVimSendMessage }
```

List of ccMail addresses who will be Carbon Copied on the message. If any of the supplied addresses are invalid the Send method will fail.

Recipients Property

```
property Recipients: TStrings; { TVimSendMessage }
```

List of ccMail addresses who will receive the message. If any of the supplied addresses are invalid the Send method will fail.

MessageTxt Property

```
property MessageTxt: String; { TVimSendMessage }
```

If this property is blank then the contents of MessageLines is sent otherwise the body text of the message is formed from the MessageTxt value. It is sent without any special formatting

MessageLines Property

```
property MessageLines: TStrings; { TVimSendMessage }
```

If the MessageTxt property is blank then the contents of MessageLines is sent as the body text of the message. It permits the sending of Memo fields.

NB Memory is allocated dynamically when passing the text to ccMail and it follows that extremely large memo fields may cause an out of memory exception

Priority Property

```
property Priority: TVimPriority; { TVimSendMessage }
```

Permitted values are

- Low
- Normal
- Urgent

Session Property

```
property Session: TVimSession; { TVimSendMessage }
```

The TVimSession object that will supply the channel to the VIM engine.

Subject Property

```
property Subject: String; { TVimSendMessage }
```

The Subject to appear in the heading of the message.

Receipt Property

```
property Receipt: Boolean; { TVimSendMessage }
```

If true then a receipt will be requested.

Registration

Distribution of registered versions of the software is via eMail only. I am unable to supply the software by mail at this point in time.

There are three versions available

VIMVCL2R.ZIP	$35 US	This is version 2 runtime only (i.e. it does not include the source files)
VIMVCL2S.ZIP	$100 US	This is version 2 runtime and full source
VIMVCL2U.ZIP	$65 US	This is version 2 source as an upgrade for registered users of the VIMVCL runtime versions 1 and 2

You may register as follows :-

Online

Via the Compuserve Software Registration forum SWREG quoting

#9609 for VIMVCL2R.ZIP

#9610 for VIMVCL2U.ZIP

#9611 for VIMVCL2S.ZIP

By Credit Card ONLY

You can order with MC, Visa, Amex, or Discover from Public (software) Library by calling 800-2424-PsL or 713-524-6394 or by FAX to 713-524-6398 or by CIS Email to 71355,470. You can also mail credit card orders to PsL at P.O.Box 35705, Houston, TX 77235-5705.

When ordering please quote the following :

PsL product id. # 14428

Whether you require the runtime/source/upgrade options

Your eMail address (failure to supply this means that your order cannot be fulfilled)

Please note that a handling charge of $1 US will be charged for orders placed via PsL.

THE ABOVE NUMBERS ARE FOR CREDIT CARD ORDERS ONLY. THE AUTHOR OF THIS PROGRAM CANNOT BE REACHED AT THESE NUMBERS.

Any questions about the status of the shipment of the order, refunds, registration options, product details, technical support, volume discounts, dealer pricing, site licenses, non-credit card orders, etc, must be directed to the author via eMail.

To insure that you get the latest version, PsL will notify us the day of your order and we will ship the product directly to you."

Support

You can contact the author

via CIS mail on 100047,123

the Internet on 100047.123@compuserve.com or mikeg@romplaza.demon.co.uk

by telephone in the UK during working hours on 01489-787709

TVimInbox

This component provides access to messages within the inbox. It must be linked to an active TVimSession component, have its own Active property set to true and either the ScanInboxOnActivate set to true or the developer must call the ScanInbox fucntion.

properties

Active

MsgFilter

ScanInboxOnActivate

Session

VimStatus

methods / fucntions / procedures

GetMessage

SaveAttachments

ScanInbox

deleteMessage

MoveMessageToFolder

SetMessageRead

run-time properties

Attachments

BCC

CC

Date

From

InboxContents

MessageLines

NewMessages

Priority

Subject

ToNames

UnreadMessages

data structures and support classes

TVimAttachment

TVimInboxMsg

TVimMsgFilter

Session property

```
property Session : TVimSession; { TVimAddressBook }
```

The TVimSession object that will supply the channel to the VIM engine.

ActiveAddressBook property

```
property ActiveAddressBook : String; { TVimAddressBook }
```

Setting this to a valid address book name causes the details of the addressees to be loaded into the Names and Gateways properties.

Names property

```
property Names : TStrings; { TVimAddressBook }
```

Holds the names from the currently active address book. In addition to the addressee name additional data may be present in the objects property of the String List - this data is stored in an object of type TVimAddress

data structures and support classes

TVimAddress

Gateways property

```
property Gateways : TStrings; { TVimAddressBook }
```

Holds details of the Post Offices and Gateways encountered when reading details from the currently active address book.

It is primarily intended for developers who wish to develop advanced address lookup functions.

AddressBookList property

```
property AddressBookList : TStrings; { TVimAddressBook }
```

Holds details of the available address books. It is populated when the Active property is set to true.

setActiveAddressBook

TVimAddress

The following Delphi definition describes the class used to store additional details of a ccMail address within a TVimAddressBook object. It is stored in the Names TStrings property as the associated object.

```
type

  TVimAddress = class(TObject)
  public
    property Comments:string;
    property PostOffice:string;
    property FullAddress:string;
    function HasChildren:boolean;
    function isGroup:boolean;
  end;
```

Comments property

```
property Comments : String; { TVimAddress }
```

The free format comments text held within the ccMail / Notes address book for the associated address.

PostOffice property

```
property PostOffice : String; { TVimAddress }
```

This is a derived field containing the 'post office' portion of the full address held within the ccMail / Notes address book. If the full address is 'Mike Gee AT HQ-PO' then the PostOffice property will be 'HQ-PO'. This is primarily for those who wish to write their own address book display routines.

FullAddress property

```
property FullAddress : String; { TVimAddress }
```

Holds the full address as held within the ccMail / Notes address book. This will usually be in the format 'Mike Gee AT HQ_PO'.

This is the value used by VIM when addressing messages.

HasChildren function

```
function HasChildren : boolean; { TVimAddress }
```

Indicates whether the address book entry is a post-office, gateway or group which points to individual user names / addresses.

isGroup function

```
property PostOffice : String; { TVimAddress }
```

Returns true if the address book entry is a mailing-list and therefore may point to additional individual user names / addresses.

Disclaimer

Limited Warranty

The SOFTWARE is distributed and licensed "AS IS". GeeSoft and D.M Gee specifically disclaim all other warranties, express or implied, including but not limited to, implied warranties of merchantability and fitness for a particular purpose, with regard to the SOFTWARE.

Liability

In no event shall GeeSoft or D.M Gee be responsible for any damages whatsoever (including but not limited to, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use or inability to use this product.

Date property

```
property Date: TDateTime; { TVimInboxMsg }
```

The date the message was despatched.

Priority property

```
property Priority: TVimPriority; { TVimInboxMsg }
```

This is the priority of the message.

MsgUnRead property

```
property MsgUnread: Boolean; { TVimInboxMsg }
```

This is true if the user has not accessed the message.

From property

```
property From : String; { TVimInboxMsg }
```

The name of the person who sent the message.

InboxContents property

```
property InboxContents: TStrings; { TVimInbox }
```

This contains details of the messages within the currently activeAddressBook. The messages may have been restricted to unread messages if the msgFilter property has been set to unreadMessages.

The ***strings*** array holds the subject of the individual messages and the associated ***objects*** array holds all additional information in a TVimInboxMsg object.

MsgFilter property

```
property MsgFilter: TVimMsgFilter; { TVimInbox }
```

This is used to restrict the messages returned by a call to the scanInbox function.

ScanInboxOnActivate

```
property ScanInboxOnActivate: Boolean; { TVimInbox }
```

If set to true the inbox will be read using the scanInbox function every time the component is activated,

Session

```
property Session: TVimSession; { TVimInbox }
```

The TVimSession object that will supply the channel to the VIM engine.

Active property

```
property Active: Boolean; { TVimInbox }
```

Setting this to true causes the inbox to be opened in preparation for the reading of messages. If the scanInboxOnActivate property is set to true then the scanInbox function will be called.

If the session associated with the component is not active then any attempt to set Active to true will fail.

VimStatus

```
property VimStatus : Boolean; { TVimInbox }
```

This is updated by the functions to reflect their completion status. It provides an alternative method of determining if a function was successfull.

Attachments property

```
property Attachments: TStrings; { TVimInbox }
```

This contains the names of message attachments in the associated ***strings*** array and the size of the attachment as a TVimAttachment in the assocoated ***objects*** array. It is populated by calls to the getMessage function.

BCC property

```
property BCC: TStrings; { TVimInbox }
```

This contains the details of the various addressees in the Blind Carbon Copy list associated with the current message. It is populated by a successful call to the getMessage function.

CC property

```
property CC: TSrings; { TVimInbox }
```

This contains the details of the various addressees in the Carbon Copy list associated with the current message. It is populated by a successful call to the getMessage function.

Date property

```
property Date: TDateTime; { TVimInbox }
```

This is the date the message was sent.

From property

```
property From: String; { TVimInbox }
```

This is the author of the current message as returned by a call to the getMessage function.

MessageLines property

```
property MessageLines : TStrings; { TVimInbox }
```

This hold the body text of the current message as returned by a call to the getMessage function.

NewMessages property

```
property NewMessages: Boolean; { TVimInbox }
```

This returns the number of new messages in the inbox and is updated each time a call is made to the scanInbox function.

Priority property

```
property Priority : TVimPriority; { TVimInbox }
```

This is the priority of the current message as returned by a successfull call to the getMessage function.

Subject property

```
property Subject : String; { TVimInbox }
```

This is the subject of the current message as returned by a call to the getMessage fucntion.

ToNames property

```
property ToNames: TStrings; { TVimInbox }
```

This is the list of addressees contained in the To portion of the current message as returned by a call to the getMessage function.

UnreadMessages property

```
property UnreadMessages: integer; { TVimInbox }
```

This returns the number of unread messages in the inbox as returned by a call to the scanInbox function. The value is not affected by the state of the msgFilter property.

GetMessage

```
function GetMessage(msgNum : integer) : Boolean; { TVimInbox }
```

This reads the message defined by ***msgNum*** which must point to a valid item within inboxContents. Upon successful completion the following properties will have been populated with data from the specified message:-

Attachments

BCC

CC

Date

From

MessageLines

Priority

Subject

ToNames

SaveAttachments

```
function SaveAttachments(attachment:integer; path:string): Boolean; { TVimInbox }
```

This will save the attachment specified by *attachment* into the path pointed to by *path*. Attachment must specify a valid entry within the Attachments property which will have been set up by a previously successful call to the getMessage function.

NB Under certain circumstances the size of the file stored will not match that specified in the associated TVimAttachment object as VIM appears to include trailing character past the EOF delimeter for text files.

ScanInbox

```
function ScanInbox: Boolean; { TVimInbox }
```

This scans the inbox and populates the inboxContents with details of the available messages. The scan may be constrained by the value of msgFilter.

There must have been a successfull call to ScanInbox prior to any calls to the getMessage function.

TVimPriority

TVimMsgFilter type

The following Delphi definition describes the type used to define the filtering of messages read from the Inbox by the ScanInbox function. It is defined in TVimInbox and is not used by any other units.

```
Type
  TVimMsgFilter = (UnReadMessages,AllMessages);
end;
```

TVimAttachment

The following Delphi definition describes the class used to store additional details of a ccMail / Notes message attachment within a TVimInbox object. It is stored in the Attachments TStrings property as the associated object.

```
Type
  TVimAttachment = class(TObject)
  public
    property Size : longInt;
  end;
```

Whats New

The following methods have been added to TVimInbox

- deleteMessage
- MoveMessageToFolder
- SetMessageRead

A source is included for the login form so that the DFM file may be customized.

TVimAddressbook now correctly recognises local user entries in the address book.

Future Plans

An additional address book component with a reduced memory overhead.

VIM addressbook aware listbox, combobox and stringgrid components which will operate independently of the current TVimAddressBook requiring a much reduced memory overhead.

Implementation of true Delphi exception events.

A 32-bit version compatible with the current Lotus beta 32-bit VIM

Lotus Notes compatibility

Access to folders.

Dynamic rather than Delphi controlled loading of VIM.DLL which will do away with the necessity for the DLLs to be on the path or in the Windows directory.

Addressing to remote gateways such as FAX and Internet servers

DeleteMessage

```
function DeleteMessage(msgNum:integer): Boolean; { TVimInbox }
```

This deletes the specified message from the inbox.

If the message is currently open as the result of a call to the getMessage function then it is closed. The message is deleted from inboxContents

MoveMessageToFolder

```
function MoveMessageToFolder(msgNum:integer; sFolder:string) : Boolean; { TVimInbox }
```

This moves the specified message from the inbox to the specified folder.

If the specified folder does not exist it is created.

If the message is currently open as the result of a call to the getMessage function then it is closed. The message is deleted from inboxContents

SetMessageRead

```
function SetMessageRead(msgNum:integer): Boolean; { TVimInbox }
```

This sets the status of the specified message to read.

Installation

Warning : VIM.DLL, MAILENG.DLL and MEMMAN.DLL must be present either in one of the directories specified in your path statement or in your Windows directory otherwise the components will not function correctly.

The software :

VIMSESS.DCU	TVimSession
VIMSEND.DCU	TVimSendMessage
VIMINBOX.DCU	TVimInbox
VIMADDBK.DCU	TVimAddressBook
VIM.DCU	Support routines
VLOGIN.DCU	Required for the ccMail login/password form
VLOGIN.DFM	The Login/Password form
VABOUT.DCU	About form
VABOUT.DFM	

Copy the files to your VCL directory and install VIMSESS.DCU, VIMSEND.DCU, VIMINBOX.DCU and VIMADDBK.DCU using Options | Install Components from the main IDE menu. They will appear on the component palette in a section called ccMail.

Help files

VIMVCL.HLP	This file
VIMVCL.KWF	Keyword file to link VIMVCL.HLP into the Delphi IDE

Copy VIMVCL.HLP to your Delphi programs directory (typically DELPHI\BIN); copy VIMVCL.KWF to your Delphi help directory (typically DELPHI\HELP) and run the HELPINST utility.