NMpop3 unit

The NMPOP3 unit contains the TNMPOP3 component and it's related types and objects.

Components
TNMPOP3

Objects
TMailMessage
TSummary

Types
TListEvent

TMailMessage object

See also Properties

Unit
NMpop3

Description
The TMailMessage object defines the structure that is used for storing E-Mail messages retrieved from the remote host.

TNMPOP3 contains an instance of TMailMessage as a property, called MailMessage, which stores the currently received E-Mail message.

See Also

TNMPOP3.MailMessage property
TNMPOP3.Summary property
TSummary Object

TMailMessage Properties

TMailMessage
Legend

- Attachments
- Body
- From
- Head
- MessageId
- Subject

Attachments property

Applies to
TMailMessage object

Declaration
```
property Attachments: TStringList;
```

Description
The Attachments property contains a string list of the filenames of files attached to the current message

Accessability: Runtime, Read-only

Body property

Applies to
TMailMessage object

Declaration
```
property Body: TStringList;
```

Description
The Body property contains the body of the current mail message.

Accessability: Runtime, Read-only

From property

Applies to
TMailMessage object

Declaration
```
property From: string;
```

Description
The From property contains the E-Mail address of the sender of the current mail message.

Accessability: Runtime, Read-only

Head property

Applies to
TMailMessage object

Declaration
```
property Head: TExStringList;
```

Description
The Head property contains the header of the current mail message.

Accessability: Runtime, Read-only

MessageId property

Applies to
TMailMessage object

Declaration
```
property MessageId: string;
```

Description
The MessageId property contains the message ID assigned to the current mail message by the mail server.

Accessability: Runtime only

Subject property

Applies to
TMailMessage object

Declaration
```
property Subject: string;
```

Description
The Subject property contains the subject line of the current E-Mail message.

Accessability: Runtime, Read-only

TSummary object

See also Properties

Unit
NMpop3

Description
The TSummary object defines the structure that E-Mail message summaries are stored in.

TNMPOP3 component contains an instance of TSummary as a property, called Summary, which contains the currently received message summary.

See also

TMailMessage Object
TNMPOP3.MailMessage property
TNMPOP3.Summary property

TSummary Properties

Bytes property

Applies to
TSummary object

Declaration
```
property Bytes: integer;
```

Description
The Bytes property contains the number of bytes in the E-Mail message the current summary is for.

From property

Applies to
TSummary object

Declaration
```
property From: string;
```

Description
The From property contains the E-Mail address for the sender of the message the current summary is for.

Header property

Applies to
TSummary object

Declaration
```
property Header: TExStringList;
```

Description
The Header propert contains the header for the E-Mail message the current summary is for.

MessageId property

Applies to
TSummary object

Declaration
```
property MessageId: string;
```

Description
The MessageId property contains the message ID assigned by the mail server for the message that is being summarized.

Subject property

Applies to
TSummary object

Declaration
```
property Subject: string;
```

Description
The Subject property contains the subject line for the current summarized E-Mail message.

 TNMPOP3 component

Heirarchy——————Properties Methods Events Tasks

Unit

NMpop3

Description

The TNMPOP3 component is used for retrieving internet E-Mail from a POP3 server. Use of this component requires a 32 bit WinSock stack, WSOCK32.DLL, which is available from many vendors, and is also include with Windows 95.

TNMPOP3 Properties
TNMPOP3
Legend

In TNMPOP3

AttachFilePath
DeleteOnRead
MailCount
MailMessage
Password
Summary
UserID

Derived from TPowersock

About
BeenCanceled
BeenTimedOut
BytesRecvd
BytesSent
BytesTotal
Connected
Handle
Host
LastErrorNo
LocalIP
Port
Proxy
ProxyPort
RemoteIP
ReplyNumber
ReportLevel
Status
TimeOut
TransactionReply
WSAInfo

Derived from TComponent

ComObject
ComponentCount

- ComponentIndex
- Components
- ComponentState
- ComponentStyle
- DesignInfo
- Owner
- Tag

VCLComObject

TNMPOP3 Methods

TNMPOP3

Legend

In TNMPOP3

- UniqueID
- DeleteMailMessage
- GetMailMessage
- GetSummary
- List
- Reset

Derived from TPowersock

- Abort
- Accept
- Cancel
- CaptureFile
- CaptureStream
- CaptureString
- CertifyConnect
- Connect
- Create
- Destroy
- Disconnect
- FilterHeader
- GetLocalAddress
- GetPortstring
- Listen
- read
- ReadLn
- RequestCloseSocket
- SendBuffer
- SendFile
- SendStream
- Transaction
- write
- writeln

Derived from TComponent

- DestroyComponents
- Destroying
- FindComponent
- FreeNotification
- FreeOnRelease
- GetParentComponent
- HasParent
- InsertComponent
- RemoveComponent
- SafeCallException

Derived from TPersistent

- Assign
- GetNamePath

Derived from TObject

- ClassInfo

ClassName
ClassNameIs
ClassParent
ClassType
CleanupInstance
DefaultHandler
Dispatch
FieldAddress
Free
FreeInstance
GetInterface
GetInterfaceEntry
GetInterfaceTable
InheritsFrom
InitInstance
InstanceSize
MethodAddress
MethodName
NewInstance

TNMPOP3 Events
TNMPOP3
Legend

In TNMPOP3

- OnAuthenticationFailed
- OnAuthenticationNeeded
- OnDecodeStart
- OnFailure
- OnList
- OnReset
- OnRetrieveEnd
- OnRetrieveStart
- OnRetriveEnd
- OnRetriveStart
- OnSuccess

Derived from TPowersock

- OnAccept
- OnConnect
  - OnConnectionFailed
- OnConnectionRequired
- OnDisconnect
- OnError
- OnHostResolved
- OnInvalidHost
- OnPacketRecvd
- OnPacketSent
- OnRead
- OnStatus

About the TNMPOP3 component

TNMPOP3 reference

Purpose

The TNMPOP3 component is used for retrieving internet E-Mail from a POP3 server. Use of this component requires a 32 bit WinSock stack, WSOCK32.DLL, which is available from many vendors, and is also include with Windows 95.

Tasks

Before E-Mail can be retrieved using the TNMPOP3 component, a connection with an E-mail server must first be established. This is accomplised by first setting the **Host** property to the mail server where the account is located, and the setting the **UserID** property and **Password** property to the user name and password of a valid E-Mail account on the server. The connection to the server is actually established after calling the **Connect** method

Retrieving Internet Mail:

By calling the **GetMailMessage** method, E-Mail can be retrieved from the mail server. The different parts of the body are stored in the **MailMessage** property.

AttachFilePath property

See also Example

Applies to
TNMPOP3 component

Declaration
```
property AttachFilePath: string;
```

Description
The AttachFilePath property specifes the directory to save any files attached to messages. By default they are saved in the same directory as the application

Scope: Published
Accessability: Runtime, Designtime

Notes:
By default attachments are saved in the same directory as application. If the directory specified as the AttachFilePath does not exist, the attachements are saved in the current directory.
If a trailing backslash ("\") is not present at the end of the AttachFilePath, one is added automatically.

See also

GetMailMessage method

Example

To recreate this example, you will need to create a new blank Delphi application.

Place 6 TEdits, a TMemo, a TCheckBox, 2 TButtons, and a TNMPOP3 on the Form.

Component Descriptions:
Edit1: Host
Edit2: User ID
Edit3: Password
Edit4: Attachment Path
Edit5: From (Person that sent the E-Mail)
Edit6: Subject of the E-Mail
Memo1: Message Body
Button1: Connect/Disconnect
Button2: Get Mail Message
CheckBox1: Delete message after reading

Insert the following code into Button1's OnClick event:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if NMPOP31.Connected then
    NMPOP31.Disconnect
  else
  begin
    NMPOP31.Host := Edit1.Text;
    NMPOP31.UserID := Edit2.Text;
    NMPOP31.Password := Edit3.Text;
    NMPOP31.DeleteOnRead := Checkbox1.Checked;
    NMPOP31.AttachFilePath := Edit4.Text;
    NMPOP31.Connect;
  end;
end;
```

When Button1 gets clicked, if there is a connection with the remote host, the **Disconnect** method is executed to close the connection. If there is no connection present, the **Host** property, which specifies the mail server to connect to, is set to the value in Edit1. The **UserID** property, which specifies the user name to log in as, is set to the vlaue in Edit2. The **Password** property, which should correspond with the UserID property, is set to the value in Edit3. If CheckBox1 is checked (true), then messages are deleted as they are retrieved by the **GetMailMessage** method, since the **DeleteOnRead** property is set to the value of CheckBox1.Checked. the **AttachFilePath** property is set to the value in Edit4, and the **Connect** method establishes a connection with the remote host.

Insert the following test into NMPOP31's OnConnect method:

```
procedure TForm1.NMPOP31Connect(Sender: TObject);
begin
  if NMPOP31.MailCount > 0 then
    ShowMessage(IntToStr(NMPOP31.MailCount)+' messages in your mailbox')
  else
    ShowMessage('No messages waiting');
end;
```

When the client connects to the mail host, the **OnConnect** event is called. If there are messages waiting on the server, a message box pops up displaying the number stored in the **MailCount** property, stating how many messages are waiting on the server. If there are no messages, a box is displayed stating that there are no messages.

Insert the following code into Button2's OnClick event:

```
procedure TForm1.Button2Click(Sender: TObject);
var
  S: String;
  M: Integer;
begin
  if NMPOP31.MailCount > 0 then
  begin
    if InputQuery('Retrieve an E-Mail message', 'Which message? (1-'+IntToStr(NMPOP31.MailCount)
+')', S) then
    begin
      M := StrToIntDef(S, -1);
      If (M < 0) or (M > NMPOP31.MailCount) then
        ShowMessage('Invalid message index')
      else
        NMPOP31.GetMailMessage(M);
    end;
  end
  else
    ShowMessage('No Messages to Get');
end;
```

When Button2 is clicked, if there are messages on the server (NMPOP31.MailCount > 0), the InputQuery function requests a message number to retrieve. If the OK button in clicked, the number typed in is checked for validity with the actual number of messages. If the specified mail number is invalid, a message box is displayed saying the message doesn't exist. If the specified message exists, it is retrieved using the **GetMailMessage** method.

Insert the following code into NMPOP31's OnRetrieveEnd event:

```
procedure TForm1.NMPOP31RetrieveEnd(Sender: TObject);
begin
  Memo1.Text := NMPOP31.MailMessage.Body.Text;
  Edit6.Text := NMPOP31.MailMessage.Subject;
  Edit5.Text := NMPOP31.MailMessage.From;
end;
```

The **OnRetrieveEnd** event is called when a message has completed being retrieved from the remote host. In this case, Memo1 is set to the **Body** of the **MailMessage** property, displaying the contents of the message.Edit6 is set to the **Subject** of the **MailMessage** property, displaying the subject line of the message. Edit5 is set to the sender of the message, stored in the **From** property of the MailMessage property

Insert the following code into NMPOP31's OnDecodeStart event:

```
procedure TForm1.NMPOP31DecodeStart(var FileName: String);
var
```

```
   S: String;
begin
   S := FileName;
   if InputQuery('Save File Attachment', 'Filename?', S) then
      FileName := S;
end;
```

When a file attachment begins decoding, the **OnDecodeStart** event is called. In this example, the InputQuery function is used to retrieve a file name. If the Ok button is clicked, the **FileName** parameter is changed to the new value entered in the input box.

DeleteOnRead property

See also Example

Applies to
TNMPOP3 component

Declaration
```
property DeleteOnRead: boolean;
```

Description
The DeleteOnRead property indicates whether to delete a message when it is read. If DeleteOnRead is set to TRUE, then messages are deleted when they are retrieved by the GetMailMessage method. Messages are left on the server after a call to GetMailMessage if DeleteOnRead is FALSE. By default they are not deleted

Default: FALSE

Scope: Published
Accessability: Run time, Design Time

Notes:
The Reset method will reset all deletion flags, so that any messages deleted via DeleteOnRead or the DeleteMailMessage method during the current session will be un-deleted.

See also

DeleteMailMessage method
GetMailMessage method

MailCount property

Example

Applies to

TNMPOP3 component

Declaration

```
property MailCount: integer;
```

Description

The MailCount property contains the number of messages on the server in the given users mailbox. This value is set after connecting to the server.

Scope: Public
Accessability: Run time

Notes:
Mailcount is set after a successful connection to the server

MailMessage property

See also Example

Applies to
TNMPOP3 component

Declaration
```
property MailMessage: TMailMessage;
```

Description
The MailMessage property is set when the GetMailMessage method is called. It contains the data that makes up the retrieved E-Mail message.

Scope: Public
Accessability: Runtime, Read-Only

Note:
The contents of the MailMessage property are changed after each call to GetMailMessage. Be sure to process the information stored in this property before making another call to GetMailMessage.

See also

Summary property
TMailMessage object

Password property

See also Example

Applies to
TNMPOP3 component

Declaration
```
property Password: string;
```

Description
The Password property specifies the password needed by the connect method to connect to a mail server.

Scope: Published
Accessability: RunTime, DesignTime

Notes:
The password given must correspond with the specified user name in the UserID property.
If the given password is invalid in any way, the OnAuthenticationFailed event is called.
If there is no password supplied, and one is needed, the OnAuthenticationNeeded event is called.

See also

OnAuthenticationFailed event
OnAuthenticationNeeded event
UserID property

Summary property

See also Example

Applies to
TNMPOP3 component

Declaration
```
property Summary: TSummary;
```

Description
The Summary property contains summary information for an E-Mail message.
To retrieve the whole E-Mail message, call the GetMailMessage method.

Scope: Public
Accessability: RunTime, ReadOnly

Notes:
This property is set after the GetSummary method is called, and contains the summary information for the last successfully retrieved message summary.
The information stored in the Summary property is reset each time a call to the GetSummary method is made, so be sure to process the information in the Summary property before subsequent calls to GetSummary are made.

See also

GetMailMessage method
GetSummary method
TSummary object

Example

To recreate this example, you will need to create a new blank Delphi application.

Place 6 TEdits, 4 TButtons, 2 TMemos, and a TNMPOP3 on the form.

***If you wish to label these components, label them in the following manner (Do not change the Name property, or the example code will not work)

Edit1 = Host
Edit2 = User ID
Edit3 = Password
Edit4 = From
Edit5 = Subject
Edit6 = Message ID
Memo1 = Status messages
Memo2 = Message Header
Button1 = Connect/Disconnect
Button2 = Summarize
Button3 = Delete
Button4 = Reset

Insert the following code into Button1's OnClick event:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if NMPOP31.Connected then
    NMPOP31.Disconnect
  else
  begin
    NMPOP31.Host := Edit1.Text;
    NMPOP31.UserID := Edit2.Text;
    NMPOP31.Password := Edit3.Text;
    NMPOP31.Connect;
  end;
end;
```

When Button1 is clicked, if there is a connection with the POP3 server, the Disconnect method is called. If there is no connection present, the Host, UserID, and Password properties are set, and the Connect method is called to establish a connection with the remote host.

Insert the following code into NMPOP31's OnConnect event:

```
procedure TForm1.NMPOP31Connect(Sender: TObject);
begin
  if NMPOP31.MailCount > 0 then
    NMPOP31.List
  else
    Memo1.Lines.Add('No Messages');
end;
```

When a connection is established with the POP3 server, if there are messages waiting in the mailbox on the server, the messages are listed with their index number and size in bytes through the use of the List method.

Insert the following code into NMPOP31's OnList event:

```
procedure TForm1.NMPOP31List(Msg, Size: Integer);
begin
  Memo1.Lines.Add('Message '+IntToStr(Msg)+': '+IntToStr(Size)+' bytes');
end;
```

When the List method is invoked, for each message on the server, the OnList event is called, passing the message index number and the size in bytes as parameters. This information is added to Memo1 for the user to view.

Insert the following code into NMPOP31's OnAuthenticationFailed event:

```
procedure TForm1.NMPOP31AuthenticationFailed(var Handled: Boolean);
var
  NewPass,
  NewID: String;
begin
  if MessageDlg('Authentication Failed. Retry?', mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  begin
    NewPass := NMPOP31.Password;
    NewID := NMPOP31.UserID;
    InputQuery('Authentication Failed','Input User ID', NewID);
    InputQuery('Authentication Failed','Input Password', NewPass);
    NMPOP31.Password := NewPass;
    NMPOP31.UserID := NewID;
    Handled := TRUE;
  end
  else
    Handled := FALSE;
end;
```

If the value of either the Password property or the UserId property are incorrect, the OnAuthenticationFailed event gets called. In this case, the user is prompted (vie the MessageDlg function) to retry. If the user clicks the yes button, the InputQuery function is called twice asking for the user's User ID and Password. The defaults are the values that were used initially before the connect. Once these values are set, the Handled parameter is set to True, which signals the client to attempt authentication again. If the user decides not to try authenticating again, the Handled parameter is set to false and the exception is raised.

Insert the following code into NMPOP31's OnAuthenticationNeeded event:

```
procedure TForm1.NMPOP31AuthenticationNeeded(var Handled: Boolean);
var
  NewPass,
  NewID: String;
begin
  NewPass := NMPOP31.Password;
  NewID := NMPOP31.UserID;
  InputQuery('Authorization Needed','Input User ID', NewID);
  InputQuery('Authorization Needed','Input Password', NewPass);
  NMPOP31.UserID := NewID;
```

```
  NMPOP31.Password := NewPass;
  Handled := TRUE;
end;
```

If either the Password property or the UserId property are left blank, the OnAuthenticationNeeded event is called. In this instance, the OnAuthenticationNeeded event prompts the user for a user ID and a password. The default values are the ones initially assigned to the UserId and Password properties so the user doesn't have to retype provided information. The handled parameter is set to true at the end of the event. If the specified information is still incomplete, an exception will be raised.

Insert the following code into Button4's OnClick event:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  NMPOP31.Reset;
end;
```

When Button4 is clicked, the Reset method gets called, resetting all delete flags for the current session. So if 3 of 4 messages on the server have been deleted with the DeleteMailMessage method, clicking Button4 will "undelete" those 3 messages.

Insert the following code into NMPOP31's OnReset event:

```
procedure TForm1.NMPOP31Reset(Sender: TObject);
begin
  Memo1.Lines.Add('Delete flags reset');
end;
```

When the Reset method is called, the OnReset event adds a line to Memo1 to inform the user that all delete flags have been reset.

Insert the following code into Button2's OnClick event:

```
procedure TForm1.Button2Click(Sender: TObject);
var
  S: String;
  M: Integer;
begin
  if NMPOP31.MailCount > 0 then
  begin
    if InputQuery('Retrieve a Summary', 'Which message? (1-'+IntToStr(NMPOP31.MailCount)+')', S)
then
    begin
      M := StrToIntDef(S, -1);
      If (M < 0) or (M > NMPOP31.MailCount) then
        ShowMessage('Invalid message index')
      else
      begin
        NMPOP31.GetSummary(M);
        Edit6.Text := NMPOP31.UniqueID(M);
      end;
    end;
  end
```

```
  else
    ShowMessage('No Messages to Summarize');
end;
```

When Button2 is clicked, the MailCount property is checked to make sure there are messages present on the mail host to retrieve summaries for. If there are messages present on the server, the InputQuery function asks the user for a message number to retrieve the summary for. This prompt also provides the minimum and maximum values acceptable. If the entered value is acceptable, the GetSummary method retrieves the summary for that message. Also, the UniqueID function retrieves the unique message ID for the selected message. If the message number specified is invalid, a message box pops up stating that the message index was invalid. If there are no messages on the server, a message box pops up stating that there are no messages to summarize.

Insert the following code into NMPOP31's OnRetrieveStart event:

```
procedure TForm1.NMPOP31RetrieveStart(Sender: TObject);
begin
  Memo1.Lines.Add('Retrieving Summary');
end;
```

When a Summary is being retrieved from the remote host, the OnRetrieveStart event notifies the user of the start of this action. This event also gets called when the **GetMailMessage** method is called.

Insert the following code into NMPOP31's OnRetrieveEnd event:

```
procedure TForm1.NMPOP31RetrieveEnd(Sender: TObject);
begin
  Memo1.Lines.Add('Summary retrieved');
  Edit4.Text := NMPOP31.Summary.From;
  Edit5.Text := NMPOP31.Summary.Subject;
  Memo2.Text := NMPOP31.Summary.Header.Text;
end;
```

When a Summary has finished being retrieved from the remote host, the OnRetrieveEnd event is called. In this instance, the OnRetrieveEnd event puts the summary information stored in the Summary property into Edit boxes and the retrieved header into Memo2. A status message is also added to Memo1 stating that the summary has been retrieved.

Insert the following code into Button3's OnClick event:

```
procedure TForm1.Button3Click(Sender: TObject);
var
  S: String;
  M: Integer;
begin
  if NMPOP31.MailCount > 0 then
  begin
    if InputQuery('Delete a Message', 'Which message? (1-'+IntToStr(NMPOP31.MailCount)+')', S) then
    begin
      M := StrToIntDef(S, -1);
      If (M < 0) or (M > NMPOP31.MailCount) then
        ShowMessage('Invalid message index')
      else
```

```
        begin
          NMPOP31.DeleteMailMessage(M);
        end;
      end;
    end
    else
      ShowMessage('No Messages to Delete');
end;
```

When Button3 is clicked, if there are any mail messages on the POP3 host, as reported by the MailCount property, the InputQuery function prompts the user for a message number in the range of 1 to the number of messages on the server. If the user specifies a valid message index, that message is marked as deleted, and will be deleted when the user disconnects if the **Reset** method is not called.

Insert the following code into NMPOP31's OnSuccess event:

```
procedure TForm1.NMPOP31Success(Sender: TObject);
begin
  ShowMessage('Message deleted.');
end;
```

When a message has been marked as deleted by the DeleteMailMessage method, the OnSuccess event signifies the success of that action. In this instance, a message box is displayed stating the deletion of the file.

Insert the following code into NMPOP31's OnFailure event:

```
procedure TForm1.NMPOP31Failure(Sender: TObject);
begin
  ShowMessage('Operation Failed');
end;
```

The OnFailure event is called when an error has occurred while either deleting an E-Mail message or when retrieving a message's ID using the UniqueID method. In this example, when an error has occured, the OnFailure event displays a message box stating that the operation has failed.

UserID property

See also Example

Applies to
TNMPOP3 component

Declaration
```
property UserID: string;
```

Description
The UserID property specifies the user name needed by the connect method to connect to a mail server.

Scope: Published
Accessability: RunTime, DesignTime

Notes:
If the supplied UserID does not have an account on the mail server specified by the **Host** property, the OnAuthenticationFailed event will be called.
If the supplied UserID and Password do not correspond, the OnAuthenticationFailed event will be called.
If there is no UserID specified, the OnAuthenticationNeeded event will be called.

See also

OnAuthenticationFailed event
OnAuthenticationNeeded event
Password property

UniqueID method

See also Example

Applies to
TNMPOP3 component

Declaration
```
function UniqueID(MailNumber: integer): string;
```

Description
The UniqueID method returns the Message Id for the message specified by the MailNumber parameter.

Parameters:
The MailNumber parameter specifies the message on the mail server to retrieve the message ID for. MailNumber corresponds to the message number, with a lower limit of 1 and an upper limit specified by the MailCount property.

Return Value:
The return value is the Message ID of the message number specified.

Scope: Public

See also

TMailMessage.MessageId property
MailCount property

DeleteMailMessage method

See also Example

Applies to
TNMPOP3 component

Declaration
```
procedure DeleteMailMessage(MailNumber: integer);
```

Description
The DeleteMailMessage method deletes the message specified by MailNumber from the mail server. If an error occurs, the OnFailure event is called and an exception is raised.
If the message is successfully marked as deleted, the OnSuccess event will be called.

Parameters:
The MailNumber parameter specified the message index number of the message to mark for deletion. This number will fall into the range of 1 to MailCount

Scope: Public

Preconditions: Must be Connected

Notes:
The Deleted mail message is not actually erased until disconnection. If the Reset method is called, all message deleted during the current session are undeleted

See also

DeleteOnRead property
MailCount property
OnFailure event
OnSuccess event
Reset method

GetMailMessage method

See also Example

Applies to
TNMPOP3 component

Declaration
```
procedure GetMailMessage(MailNumber: integer);
```

Description
The GetMailMessage method gets the mail message specified by MailNumber and stores its data in the MailMessage property. If an error occurs, an exception is raised.

Parameters:
The MailNumber parameter is the index of the mail message to retrieve.
The range for this value is 1 to MailCount.

Scope: Public

Notes:
You must be connected to call this method. If a connection is not present, the OnConnectionRequired event is called.
When retrieval begins, the OnRetrieveStart (OnRetriveStart) event is called.
When retrieval completes, the OnRetrieveEnd (OnRetriveEnd) event is called.
If the message is retrieved successfully, the OnSuccess event is called.
If DeleteOnRead is set to true, calling GetMailMessage will also mark the retrieved message as deleted.

See also

DeleteOnRead property
MailCount property
OnFailure event
OnRetrieveEnd event
OnRetriveEnd event
OnRetrieveStart event
OnRetriveStart event
OnSuccess event
Reset method

GetSummary method

See also Example

Applies to
TNMPOP3 component

Declaration
```
procedure GetSummary(MailNumber: integer);
```

Description
The GetSummary method retrieves the message summary for the message specified by MailNumber, and stores the message summary data in the Summary property. If an error occurs, an exception is raised.

Parameters:
The MailNumber parameter specifies the message number that the summary will be retrieved for.

Scope: Public

Notes:
You must be connected to call this method. If a connection is not present, the OnConnectionRequired event is called.
When retrieval begins, the OnRetrieveStart (OnRetriveStart) event is called.
When retrieval completes, the OnRetrieveEnd (OnRetriveEnd) event is called.
If the message summary is retrieved successfully, the OnSuccess event is called.

See also

MailCount property
OnFailure event
OnRetrieveEnd event
OnRetriveEnd event
OnRetrieveStart event
OnRetriveStart event
OnSuccess event

List method

See also Example

Applies to
TNMPOP3 component

Declaration
```
procedure List;
```

Description
The List method retrieves a list of message numbers and the size of each message. For each message listed, the OnList event is called, passing the message number and size as parameters. If an error occurs, an exception is raised.

Scope: Public

Notes:
You must be connected to call this method. If a connection is not present, the OnConnectionRequired event is called.

See also

OnList event

Reset method

See also Example

Applies to

TNMPOP3 component

Declaration

```
procedure Reset;
```

Description

The Reset method undeletes any mail messages marked Deleted by the DeleteMailMessage method or by calling GetMailMessage with DeleteOnRead set to true during the current session. If an error occurs, an exception is raised.

Scope: Public

Notes:

You must be connected to call this method. If a connection is not present, the OnConnectionRequired event is called.

If the reset is successful, the OnReset event is called.

See also

DeleteMailMessage method
DeleteOnRead property
OnReset event

OnAuthenticationFailed event

See also Example

Applies to

TNMPOP3 component

Declaration

```
property OnAuthenticationFailed: THandlerEvent;
```

Description

The OnAuthenticationFailed event is called when the POP3 server requires a UserID and password to log in, and one is not provided, or if the provided UserID/Password pair is invalid. If Handled is set to TRUE, then the connection/Authentication is attempted again, if Handled is FALSE (the default), then an exception is raised, and the connection attempt is canceled.

See also

OnAuthenticationNeeded event
Password property
UserID property

OnAuthenticationNeeded event

See also Example

Applies to

TNMPOP3 component

Declaration

```
property OnAuthenticationNeeded: THandlerEvent;
```

Description

The OnAuthenticationNeeded event is called when a UserID and Password are required, but one or both are not provided. If the Handled parameter is set to TRUE, then the user ID and password are attempted again. If Handled is FALSE, then an exception is raised. Handled is FALSE by default.

See also

OnAuthenticationFailed event
Password property
UserID property

OnFailure event

See also Example

Applies to

TNMPOP3 component

Declaration

```
property OnFailure: TNotifyEvent;
```

Description

The OnFailure event is called when an operation fails. This includes:

- Deleting a mail message
- Retrieving a message's Message ID

See also

DeleteMailMessage method
UniqueID method

OnList event

See also Example

Applies to
TNMPOP3 component

Declaration
```
property OnList: TListEvent;
```

Description
The OnList event is fired after receiving the size and message number of each message when the **List** method is called.

Parameters:
The Msg parameter is the ID number of the message, which gets passed to methods such as GetMailMessage and DeleteMailMessage. The Size parameter is the size of the message in bytes.

See also

List method

OnReset event

See also Example

Applies to

TNMPOP3 component

Declaration

```
property OnReset: TNotifyEvent;
```

Description

The OnReset event is called after a successfull call to the **Reset** method. The calling of this event signifies that all messages marked as deleted during this session are no longer marked as deleted.

See also

DeleteMailMessage method
DeleteOnRead property
Reset method

OnRetrieveEnd event

See also Example

Applies to
TNMPOP3 component

Declaration
```
property OnRetrieveEnd: TNotifyEvent;
```

Description
The OnRetrieveEnd event is called when a mail message or message summary completes retrieval.

Notes:
This event replaces the previously published event named **OnRetriveEnd**. The incorrectly spelled event is still available as Public so that applications using the previous version of this control will still function.

See also

GetMailMessage method
GetSummary method
OnRetriveEnd event
OnRetrieveStart event
OnRetriveStart event

OnRetrieveStart event

See also Example

Applies to
TNMPOP3 component

Declaration
```
property OnRetrieveStart: TNotifyEvent;
```

Description
The OnRetrieveStart event is called when a message begins retrieval.

Notes:
This event replaces the previously published event named **OnRetriveStart**. The incorrectly spelled event is still available as Public so that applications using the previous version of this control will still function.

See also

GetMailMessage method
GetSummary method
OnRetriveEnd event
OnRetrieveEnd event
OnRetriveStart event

OnRetriveEnd event

Declaration
```
property OnRetriveEnd: TNotifyEvent;
```

Description
The OnRetriveEnd event is identical to the OnRetrieveEnd event in all ways other than its spelling. This incorrectly spelled event is provided for backward compatability.

Notes:
For information on this event, see the OnRetrieveEnd event.

OnRetriveStart event

Declaration
```
property OnRetriveStart: TNotifyEvent;
```

Description
The OnRetriveStart event is identical to the OnRetrieveStart event in all ways other than its spelling. This incorrectly spelled event is provided for backward compatability.

Notes:
For information on this event, see the OnRetrieveStart event.

OnSuccess event

See also Example

Applies to

TNMPOP3 component

Declaration

```
property OnSuccess: TNotifyEvent;
```

Description

The OnSuccess event is called when a message has been successfully marked as deleted by the **DeleteMailMessage** method.

See also

DeleteMailMessage method

TListEvent type

Unit
NMpop3

Declaration
```
type
  TListEvent = procedure(Msg, Size: integer) of object;
```

Description
The TListEvent type is used for the OnList event. It passes 2 integer parameters. In the case of the OnList event, the Msg parameter specifies the message number of the message being listed, and the Size parameter specifies the size in bytes of that message.

Legend

- Run-time only
- Read-Only
- Published
- Protected
- Key item

Heirarchy

TObject
|
TPersistent
|
TComponent
|
TPowersock

TVarFileNameEvent Type

Unit
NMpop3

Declaration
```
type
  TVarFileNameEvent = procedure(var FileName: String) of object;
```

Description
The TVarFileNameEvent type is used for events where a changeable filename is required.

OnDecodeStart event

See also Example

Applies to

TNMPOP3 component

Declaration

```
property OnDecodeStart: TVarFileNameEvent;
```

Description

The OnDecodeStart event is called when a file attachment is about to be decoded and saved to disk.

Event Parameters:

The **FileName** parameter specifies the name of the file attachment. This value can be changed to change the name of the file attachment.

Notes:

This event does not get called if there are no file attachments.
Do not provide a path in addition to the filename, because the **AttachFilePath** property specifies the directory to save attached files into.

See Also

AttachFilePath property
Attachments property