TRUEyesSENDMAILyesyesyesUsing the SendMail ComponentSENDMAILyesyes15/06/95

Table of Contents

Using The SendMail Component

Properties Methods Events Application Notes

The *StarTech* SendMail component is used to send Internet Mail. It allows the user to specify multiple message recipients and attach documents to the mail message in a MIME compliant fashion. Before sending mail, several properties must be set. First, the properties of the SMTP (Small Mail Transfer Protocol) server that will be used to send mail must be set, using the SMTP_Server and SMTP_Port properties.

Next, the recipient of the e-mail message should be specified using the TO_Name and TO_Address properties. You should also specify the name of the sender, using the FROM_Name property, as well as the reply address, using the FROM_Address property.

If you want your message to include a Subject: header, set the Subject property. Additional headers can be specified using the Headers Property. If you want copies of your message to be sent to more than one address, specify those addresses using the ccList property.

Enter your mail message using the MailText property. If you want to send any attachments with your message, using the Attachments property to specify the file names of these attachments. You are now ready to send your message. This is done by setting the Action property. Progress and any errors will be indicated by the OnMailInfo event and the Status property

The Close method is only used to cancel the sending of mail.

SMTP_Server Property

Applies to:
TSendMail

Declaration:
property SMTP_Server: String;

Description:
The SMTP_Server property is used to set the address of the Small Mail Transfer Protocol server that is to be used to send mail. It can either be the name of the server (for example mail.acme.com) or the address of the server (197.65.4.12).

See also:
SMTP_Port Property

The SMTP_Port Property

Applies to:
TSendMail

Declaration:
property SMTP_Port: Integer;

Description:
The SMTP_Port property is used to set the port number of the Small Mail Transfer Protocol server that is to be used to send mail. Unless an unusual setup is used, the port number of an SMTP server is usually 25.

See also:
SMTP_Server property

TO_NAME Property

Applies to:
TSendMail

Declaration:
property TO_Name: String;

Description:
The TO_Name property is used to set the TO: header in your e-mail message. The value of this property is not critical to the sending of the message, but it must not be blank.

See also:
TO_Address Property

TO_Address Property

Applies to:
TSendMail

Declaration:
property TO_Address: String;

Description:
The TO_Address property is used to specify the mail address of the recipient. This property is critical to the proper delivery of the message. An Internet mail address is composed of a user name, followed by @, followed by the system name, with no intervening spaces. For example, the address of user *jsmith* on system *acme.com* is *jsmith@acme.com*

See also:
TO_Name Property
ccList Property

FROM_Name Property

Applies to:
TSendMail

Declaration:
property FROM_Name: String;

Description:
The FROM_Name property is used to set the FROM: header in the e-mail message. The value of this property is not critical, but it must not be blank.

See also:
FROM_Address property

FROM_Address Property

Applies to:
TSendMail

Declaration:
property FROM_Address: String;;

Description:
The FROM_Address property is used to set the mail address of the sender. It must be set correctly so that replies to the message can be sent to the correct address.

See also:
FROM_Name Property

Subject Property

Applies to:
TSendMail

Declaration:
property Subject: String;

Description:
If you want your message to contain a Subject: header, set the Subject property. The setting of this property is optional.

Attachments Property

Applies to:
TSendMail

Declaration:
property Attachments: TStrings;

Description:
If you want to attach files to your mail message, use the Attachments property. The Attachments property is of type Tstrings, so any method that applies to Tstrings applies to Attachments, most notably Clear, Add and Assign. Use Attachments.Clear to clear the list of attachments between messages. Use Attachments.Add() to add attachments one at a time, or use Attachments.Assign() to copy the contents of a list box Items to the attachments.list. Attachments should be specified using the full path name of the file. For example, to attach the *chimes.wav* file (from the *windows* directory on *C:* drive), you would code :

```
Atttachments.Add(c:\windows\chimes.wav);
```

See also:
MailText Property

Action Property

Applies to:
TSendMail

Declaration:
property Action: TMailAction;
type TMailAction=(None,Send_Mail);

Description:
Setting Action to Send_Mail will send the mail message. You should not send the mail message until you have set all other properties regarding the destination and content of the message. After setting Action to Send_Mail, you should check the Status property for any errors. See the Application Notes for more details.

See also:
Status Property
Application Notes

ccList Property

Applies to:
TSendMail

Declaration:
property ccList: TStrings;

Description:
If you want to send copies of your message to more than one recipient, use the ccList property. The ccList property is of type Tstrings, so any method that applies to Tstrings applies to ccList, most notably Clear, Add and Assign. Use ccList.Clear to clear the list of additional recipients between messages. Use ccList.Add() to add recipientsaddresses one at a time, or use ccList.Assign() to copy the contents of a list box Items to the additional recipients.list. Additional recipients should be specified using the e-mail address of the recipient. For example, to send a copy of your message to user *janedoe* on system *acme.com*, you would code :

ccList.Add(janedoe@acme.com);

See also:
TO_Address property

MailText Property

Applies to:
TSendMail

Declaration:
property MailText: TStrings;

Description:
You will use the MailText property to set the content of your message. The MailText property is of type Tstrings, so any method that applies to Tstrings applies to MailText, most notably Clear, Add and Assign. Use MailText.Clear to clear the message text between messages. Use MailText.Add() to add message lines one at a time, or use MailText.Assign() to copy the contents of a Memo Items to the message text.

See also:
Attachments Property

OnMailInfo Event

Applies to:
TSendMail

Declaration:
property OnMailInfo: TMailInfoEvent;

TMailInfoEvent = procedure (Sender : TObject; info: MailInfo; addinfo: string) of object;

type MailInfo=(ServerConnected,ServerDisconnected,SendingMessage,
TraceIn,TraceOut,NoWinsock,NoMailServer,MailDestinationError,
MailOriginError,InvalidServer,SocketError,ReadError,WriteError,
ProtocolError,BadAttachment,AttachmentName,AttachmentSize,
AttachmentBytes);

Description:
The OnMailInfo event indicates the progress of the sending of the mail message, returning with the *info* parameter of type MailInfo and a string *addinfo* containing additional information. Refer to MailInfo for possible values.

See also:
MailInfo Values

MailInfo Values

Applies to:
TSendMail

Declaration:
type MailInfo=(ServerConnected,ServerDisconnected,SendingMessage,
TraceIn,TraceOut,NoWinsock,NoMailServer,MailDestinationError,
MailOriginError,InvalidServer,SocketError,ReadError,WriteError,
ProtocolError,BadAttachment,AttachmentName,AttachmentSize,
AttachmentBytes);

Description:
Values that can be returned in the *info* parameter of the OnMailInfo event handler are:

- **ServerConnected:** The component has connected successfully to a server. *addinfo* will contain the numerical address of the server.
- **Server Disconnected:** The server has closed the connection, either due to the completion of the transaction, or at the request of the component following an error.
- **SendingMessage:** One line of the message text has been sent. *addinfo* will contain the line number.
- **TraceIn:** Traces everything received from the server. This not normally used in applications, unless you want to follow the transaction step by step. *addinfo* will contain the line received from the server.
- **TraceOut:** Traces everything sent by the component to the server. This not normally used in applications, unless you want to follow the transaction step by step. *addinfo* will contain the line sent to the server.
- **SocketError,ReadError,WriteError:** An internal error occurred in the Winsock interface. The connection is automatically close. This should happen very rarely.
- **Protocol Error:** An unexpected response was received from the server. The connection will be closed. This will happen, for instance, if one connects to a server at port 79 (FTP) instead of port 25 (SMTP).
- **BadAttachment:** One of the files specified in the Attachments property does not exist. The message will continue to be sent.
- **AttachmentName:** An attachment is about to be sent. *addinfo* will contain the filename.
- **AttachmentSize:** The size of the attachment is being sent in *addinfo.*
- **AttachmentBytes:** The number of attachment bytes sent so far is contained in *addinfo.*
- **Available: T**he transaction has completed.
- **PrematureDisconnect:** Disconnected before completion of transaction

The following values can be found in the Status property:

- **NoWinsock:** Generated at component creation when Winsock is not installed.
- **NoMailServer:** The SMTP_Server property is not set.
- **MailDestinationError:** The TO_Address property is not set.
- **MailOriginError:** The FROM_Address property is not set.
- **InvalidServer:** The address or name specified in SMTP_Server could not be resolved.
- **SocketError,ReadError,WriteError:** An internal error occurred in the Winsock interface. The connection is automatically close. This should happen very rarely.
- **Protocol Error:** An unexpected response was received from the server. The connection will be closed. This will happen, for instance, if one connects to a server at port 79 (FTP) instead of port 25 (SMTP).
- **BadAttachment:** One of the files specified in the Attachments property does not exist. The message will continue to be sent.
- **Available:** The transaction completed successfully.
- **Busy:** The transaction is proceeding normally
- **AlreadyBusy:** Cannot start a new transaction, still working on last transaction.

- **PrematureDisconnect:** Disconnected before completion of transaction

See also:

OnMailInfo Event
Status Property

Close Method

Applies to:
TSendMail

Declaration:
procedure Close;

Description:
You should provide a way to cancel the sending of a mail message, in case the mail server gets hung for whatever reason. This could take to form of a cancel button which calls the Close procedure.

Headers Property

Applies to:
TSendMail

Declaration:
property Headers: TStrings;

Description:
The Headers property allows the user to specify custom headers. The Headers property is of type Tstrings, so any method that applies to Tstrings applies to Headers, most notably Clear, Add and Assign. Use Headers.Clear to clear the headers between messages. Use Headers.Add() to add headers one at a time, or use Headers.Assign() to copy the contents of a TStrings to the Headers property.
In keeping with RFC 822, non standard headers should be prefaced with X-. For example:

X-MyCustomHeader: This is a sample header.

Also note that a header name must be followed by a :, or it will be confused as message text by many mail readers.

Action	MailText
Attachments	SMTP_Port
ccList	SMTP_Server
FROM_Address	Subject
FROM_Name	TO_Address
Headers	TO_Name
Status	

OnMailInfo

Close

Application Notes

The following code fragments show the minimal steps that need to be taken to send a message and check for errors.

Step 1. Setup the Mail Information:

```
SendMail.FromAddress:='jsmith@acme.com';
SendMail.ToAddress:='bgates@microsoft.com';
etc...
```

Step 2.Send the Message:

```
SendMail.Action:=Send_Mail;
if SendMail.Status=Busy then
        everything ok so far...
else
        Status contains an error.transaction is over.
```

Step 3:. Wait for completion:

```
procedure SendMailMailInfo(sender:Tobject,info:MailInfo,addinfo:string);
begin
        if info=Available then
        begin
                if Status=Available then
                        message was sent with no errors;
                else
                        Status has error, message was not sent successfully.
                end;
        end;
end;
```

The Status Property

Applies to:

TSendMail

Declaration:

property Status: MailInfo;

Description:

The Status property reports error conditions at the beginning and end of mailing transactions. It should be checked after setting the Action property for one of the following values: Busy, AlreadyBusy, NoWinsock, NoMailServer, MailDestinationError, MailOriginError, InvalidServer, SocketError, which are described in MailInfo values. Any value of Status other than Busy indicates an error condition.

The Status property should also be checked after receiving a MailInfo event with an info value of Available. Status should then be Available, if it is not then Status will contain an error condition. See the Application Notes for an example of using the Status property to check for errors.

See also:

Action Property
OnMailInfo Event
Application Notes

template

Applies to:
TSendMail

Declaration:
property;

Description:

See also: