NMMSG unit

The NMMSG unit contains the TNMMsg component for sending simple messages across the internet or an intranet, and the TNMMsgServ component, which is used for receiving messages sent by the TNMMsg component.

Purpose of the FastNet Messaging Components
The main purpose of the components in this unit is to provide an example for you, the user, so that you may use the base classes provided with the FastNet Tools for Delphi to solve custom internet/intranet development problems with custom internet protocols. You may find the TNMMsg and TNMMsgServ components useful to you, but by examining the source code, you will discover how to use the base classes, TPowersock and TNMGeneralServer, to create your own components to more specifically address your needs.

Components
TNMMsg
TNMMSGServ

Types
TMSGEvent

TNMMsg component

Heirarchy Properties Methods Events Tasks

Unit
NMMSG

Description

The TNMMsg component is used for sending simple ASCII text messages across the internet or an intranet using the TCP/IP protocol. The host that the message is being sent to should be running a server created from the TNMGeneralServer component.

TNMMsg Properties
TNMMsg
Legend

In TNMMsg
FromName

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

TNMMsg Methods

TNMMsg
Legend

In TNMMsg

- PostIt

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

TNMMsg Events
TNMMsg
Legend

In TNMMsg
OnMessageSent

Derived from TPowersock
- OnAccept
- OnConnect
  - OnConnectionFailed
- OnConnectionRequired
- OnDisconnect
- OnError

OnHostResolved
OnInvalidHost
OnPacketRecvd
OnPacketSent
OnRead
OnStatus

About the TNMMsg component

TNMMsg reference

Purpose

The purpose of the TNMMsg component is to send simple messages across the internet or an intranet. It must be paired with the TNMMsgServ component to provide 2-way communication.The use of this component requires a 32-bit Winsock stack, WSOCK32.DLL, which is available from many vendors, and is included with Windows 95.

Tasks

Before you can send a message, you must first set the **Host** property to the remote host you wish to send the message to. You must also set the **Port** property to correspond with the port of the remote message host. Set the **FromName** property so that the recipients of the message will know who it is coming from. Once these properties are set, you can send messages by calling the **PostIt** method.

FromName property

See also Example

Declaration

```
property FromName: string;
```

Description

The FromName property notifies the recipient who the sender of the message is.

Scope: Published
Accessibility: DesignTime, RunTime

See also

PostIt method

Example

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

Place 2 TEdits, 2 TMemos, a TNMMsg, and a TNMMsgServ on the form.

Set the **Host** property of NMMsg1 to 127.0.0.1 This is the IP address for localhost. If you later wish to test this example across computers, set the Host property to the IP address of the computer you wish to communicate with.

Insert the following code into Edit1's OnKeyPress event:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then
  begin
    NMMsg1.FromName := Edit2.Text;
    NMMsg1.PostIt(Edit1.Text);
  end;
end;
```

When the key pressed in Edit1 is #13 (a carriage return, the first character sent when you press ENTER), the **FromName** property of NMMsg1 is set to the value of Edit2.Text. The message entered in Edit1 is then sent to the remote host via the **PostIt** method.

Insert the following code into NMMsg1's OnMessageSent event:

```
procedure TForm1.NMMsg1MessageSent(Sender: TObject);
begin
  Memo2.Lines.Add('Message Sent');
end;
```

This is the event handler for the **OnMessageSent** event. This simple adds the words Message Sent to Memo2, which is serving as a sort of status window.

Insert the following code into NMMsgServ1's OnMSG event:

```
procedure TForm1.NMMSGServ1MSG(Sender: TComponent; const sFrom, sMsg: String);
begin
  Memo2.Lines.Add('Incoming message from '+sFrom);
  Memo1.Lines.Add('['+sFrom+'] '+sMsg);
end;
```

This is the event handler for the TNMMsgServ **OnMSG** event. When a message comes in, this event adds a line to Memo2 stating who the message is from. Memo1 is actually updated to include the new message and who it is from.

Running the Application:
When the application is run, simply type your name into Edit2. Type your message into Edit1, and press ENTER when you are finished. The message will be displayed in Memo1.

PostIt method

See also Example

Declaration

```
function PostIt(const sMsg: string): string;
```

Description

The PostIt method sends the message specified by the sMsg parameter to the remote msg host. It returns an OK from the server if the send is successful.

Parameters:
The sMsg parameter specifies the message to be sent to the remote msg host.

Return Value:
The return value from this method is an OK from the server if the message has been received successfully.
If the message is sent successfully, the **OnMessageSent** event is called.

Notes:
The **Host** property must be set to a valid host name or IP address, and there must be a server running on the port specified by the **Port** property. This remote host must accept the connection, message, and return an OK so the client can disconnect.
The **FromName** property should be set before you call the PostIt method

See also

FromName property
OnMessageSent event

OnMessageSent event

See also Example

Declaration
```
property OnMessageSent: TNotifyEvent;
```

Description
The OnMessageSent event is called when a message is sent successfully to the remote message host.

Notes:
This event is called when the **PostIt** method is invoked successfully.

See also

PostIt method

TNMMSGServ component

Heirarchy Properties Methods Events Tasks

Unit
NMMSG

Description
The TNMMsgServ component is used to received messages sent with the TNMMsg component. Rather than just use the TNMMsg and TNMMsgServ components, you should try creating your own custom protocol to address the problem(s) you are trying to solve.

TNMMsgServ Properties
TNMMsgServ
Legend

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

TNMMsgServ Methods
TNMMsgServ
Legend

Derived from TNMGeneralServer
- Serve

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

TNMMsgServ Events
TNMMsgServ
Legend

In TNMMsgServ
- OnMSG

Derived from TNMGeneralServer
- OnClientContact

Derived from TPowersock
OnConnect
OnConnectionFailed
OnDisconnect
OnHostResolved
OnInvalidHost
OnPacketRecvd
OnPacketSent
OnRead
OnStatus

About the TNMMSGServ component

TNMMSGServ reference

Purpose

The purpose of the TNMMsgServ component is to receive messages sent with the TNMMsg component. The use of this component requires a 32-bit Winsock stack, WSOCK32.DLL, which is available from many vendors, and is included with Windows 95.

Tasks

While you are designing your application, set the **Port** property in the object inspector to the port that your message server will listen on, if you wish it to differ from the default.
To receive and process messages sent to the message server, write an event handler for the **OnMSG** event.

OnMsg event

Example

Declaration

```
property OnMSG: TMsgEvent;
```

Description

The OnMSG event is called when the client has sent a message to the server.

Parameters:

The sFrom parameter specifies who the message is from.
The sMsg parameter is the message itself.

TMSGEvent type

Unit
NMMSG

Declaration
```
type TMSGEvent = procedure(Sender: TComponent; const sFrom, sMsg: string) of object;
```

Description
The TMSGEvent type is used for the OnMSG event.

Parameters
The sFrom parameter specifies who the message is from.
The sMsg parameter is the message itself.

Legend

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

Heirarchy

TObject
|
TPersistent
|
TComponent
|
TPowersock

Heirarchy

TObject
|
TPersistent
|
TComponent
|
TPowersock
|
TNMGeneralServer