NMUDP unit

The NMUDP unit contains TNMUDP and it's related types and constants.

Components
TNMUDP

Types
TBuffInvalid
THandlerEvent
TOnErrorEvent
TOnReceive
TOnStatus
TStreamInvalid

TNMUDP component

Heirarchy Properties Methods Events Tasks

Unit

NMUDP

Description

The TNMUDP component is used for implementing the User Datagram Protocol (UDP) for sending datagram packets across the internet or an intranet.

TNMUDP Properties

TNMUDP

Legend

In TNMUDP

- LocalPort
- 🔑 RemoteHost
- 🔑 RemotePort
- ReportLevel

Derived from TComponent

- ▶ ComObject
- ▶ ComponentCount
- ComponentIndex
- ▶ Components
- ▶ ComponentState
- ▶ ComponentStyle
- DesignInfo
- ▶ Owner
- 🟩 Tag
- VCLComObject

TNMUDP Methods

TNMUDP
Legend

In TNMUDP

- Create
- Destroy
- 🔑 ReadBuffer
- 🔑 ReadStream
- 🔑 SendBuffer
- 🔑 SendStream

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

TNMUDP Events

TNMUDP

Legend

In TNMUDP

OnBufferInvalid

🔑 OnDataReceived

OnDataSend

OnInvalidHost

OnStatus

OnStreamInvalid

About the TNMUDP component

TNMUDP reference

Purpose

The purpose of the TNMUDP component is for sending datagram packets across the internet or an intranet using the User Datagram Protocol (UDP).

RFC: RFC 768

Tasks

Sending UDP Data
Before you can send datagram packets using UDP, you need to know the remote host and remote port you will be sending data to. Assign the remote host to the **RemoteHost** property, and the remote port to the **RemotePort** property.

To actually send the data, you can use either the **SendBuffer** method, for sending buffers (arrays of characters) to the remote host, or the **SendStream** method, for sending streams of data.

Receiving UDP Data
Before you can receive UDP data, you must set the **LocalPort** property. This property must be set at design-time, and cannot be changed during runtime once set.

When there is UDP data available to be read, the **OnDataAvailable** event is called. Within this event, use the **ReadBuffer** method to read the data into a buffer, or the **ReadStream** method to read the data into a stream.

LocalPort property

See also Example

Applies to

TNMUDP component

Declaration

```
property LocalPort: integer;
```

Description

The LocalPort property specifies a port number to listen for datagram packets sent to it on.

Note:

This property should not be set to 0 (zero) if you wish to receive UDP data.
Also note, this property can only be set during design-time, **unless the component is created dynamically in your source code (using the **Create** method), in which case the **LocalPort** property can be set **once**.

Scope: Published
Accessability: Runtime**, design-time

See also

RemotePort property

Example

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

Place 2 TMemos, a TButton, and a TNMUDP on the form.

Memo1: Window for receiving data
Memo2: Status window
Button1: Sends UDP Data
NMUDP1: client and server for sending and receiving data

Insert the following code into Button1's OnClick event:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  C: Array [1..3] of Char;
begin
  C := 'cat';
  NMUDP1.RemoteHost := '127.0.0.1';
  NMUDP1.ReportLevel := Status_Basic;
  NMUDP1.LocalPort := 6668;
  NMUDP1.RemotePort := 6668;
  NMUDP1.SendBuffer(C, 3);
end;
```

When Button1 is clicked, C (a variable of an array of characters, 3 to be exact) is filled with the value cat. The **RemoteHost** property is set to 127.0.0.1, which is the IP address for local host. This could just as easily be the host name or IP address of a remote computer as well. The **ReportLevel** property is set to Status_Basic, to provide only basic status messages in the **OnStatus** event. The **LocalPort** property is set to 6668, so that any data sent to the computer running this application on port 6668 will be received by this component. The remote port property is also set to 6668, for sending data to port 6668 of 127.0.0.1 (basically, sending data to itself). The buffer C is now sent using the **SendBuffer** method.

Insert the followint code into NMUDP1's OnBufferInvalid event:

```
procedure TForm1.NMUDP1BufferInvalid(var handled: Boolean; var Buff: array of Char; var length:
Integer);
begin
  ShowMessage('Buffer Invalid: Buffer contains no data');
end;
```

When the **OnInvalidBuffer** event is called, a message is displayed to the user informing them that the buffer being sent is invalid because it contains no data. This error could be corrected by modifying the **Buff** parameter so it contains the data to be sent, and the **length** parameter to contain the length of the Buffer. The **handled** property would then have to be set to TRUE to allow the component to send the data again.

Insert the following code into NMUDP1's OnDataReceived event:

```
procedure TForm1.NMUDP1DataReceived(Sender: TComponent; NumberBytes: Integer; FromIP:
String; Port: Integer);
var
  C: array [1..3] of Char;
  I: Integer;
```

```
begin
  if NumberBytes <= 3 then
  begin
    NMUDP1.ReadBuffer(C, I);
    Memo1.Lines.Add(C+': received '+IntToStr(I)+' bytes from '+FromIP+' on port '+IntToStr(Port));
  end
  else
    Memo1.Lines.Add(IntToStr(I)+' bytes incoming, buffer too small');
end;
```

When data is received by NMUDP1, if the **NumberBytes** parameter is 3 or less (3 or les bytes), the data is read into an array of characters (C) by the **ReadBuffer** method and displayed in Memo1, along with how many bytes were actually read, the IP address of the computer sending the data (**FromIP**), and the port the data was sent from (**Port** parameter). If there are more than 3 bytes, Memo1 is updated to inform the user that the incoming data was too large for the supplied buffer.

Insert the following code into NMUDP1's OnDataSend event:

```
procedure TForm1.NMUDP1DataSend(Sender: TObject);
begin
  Memo2.Lines.Add('Data sent');
end;
```

When data has been successfully sent by pressing Button1, the **OnDataSend** event is called, and adds a status line stating the data was sent to Memo2.

Insert the following code into NMUDP1's OnStatus event:

```
procedure TForm1.NMUDP1Status(Sender: TComponent; status: String);
begin
  Memo2.Lines.Add(status);
end;
```

When a status message is received, the **OnStatus** event adds the current status string to Memo2.

Insert the following code into NMUDP1's OnInvalidHost event:

```
procedure TForm1.NMUDP1InvalidHost(var handled: Boolean);
var
  S: String;
begin
  S := NMUDP1.RemoteHost;
  if InputQuery('Invalid host', 'Specify valid hostname: ', S) then
  begin
    NMUDP1.RemoteHost := S;
    handled := TRUE;
  end;
end;
```

When the host name specified to send data to is an invalid host name or IP address, the **OnInvalidHost** event is called. In this instance, the **InputQuery** function gives the user the opportunity to correct the invalid name. If the user clicks the Ok button, the host name entered is set as the host to send data to, and the **handled** parameter is set to true, which allows the component to attempt the action again. If the user clicks the Cancel button, the host is not changed, and handled remains false, raising an exception.

Example Description:
This simple example sends data to itself using a single TNMUDP component that acts as both a client and a server. When Button1 gets clicked, the data is sent to the local machine. The data is then received by the same component, and manipulated accordingly.

OnBufferInvalid event

See also Example

Applies to

TNMUDP component

Declaration

```
property OnBufferInvalid: TBuffInvalid;
```

Description

The OnBufferInvalid event is called if a buffer sent to the **SendBuffer** method contains no data.

Event Parameters:

The **Buff** parameter is the buffer to be sent, and the **length** parameter is the size of the buffer. These parameter may be modified to reflect a valid buffer. If the buffer is modified to be valid, the **handled** parameter can be set to TRUE, and the buffer will resend. If the handled parameter is FALSE (default), and exception is raised.

See also

OnStreamInvalid event
SendBuffer method

OnDataReceived event

See also Example

Applies to

TNMUDP component

Declaration

```
property OnDataReceived: TOnReceive;
```

Description

The OnDataReceived event is called when data is received from the remote host

Event Parameters:

The **NumberBytes** parameter specifies the number of incoming bytes.
The **FromIP** parameter specifies the IP address of the computer that sent the data.
The **Port** property specifies which port the data was sent from.

See also

ReadBuffer method
ReadStream method

OnStatus event

See also Example

Applies to

TNMUDP component

Declaration

```
property OnStatus: TOnStatus;
```

Description

The OnStatus event is called when there is a status change in the component.

Event Parameters:

The **status** parameter is the current status of the component.

See Also

ReportLevel property

OnStreamInvalid event

See also Example

Applies to
TNMUDP component

Declaration
```
property OnStreamInvalid: TStreamInvalid;
```

Description
The OnStreamInvalid event is called when a steam passed to the **SendStream** method contains no data.

Event Parameters:
The **Stream** parameter is the stream that contains no data. It should be populated with data, and the **handled** parameter should be set to TRUE. If the **handled** parameter is TRUE, the stream is resent. If handled is FALSE, an exception is raised.

See also

OnBufferInvalid event
SendStream method

Example

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

Place a TButton, 2 TMemos, a TEdit, and a TNMUDP on the form.

Insert the following code into Button1's OnClick event:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  MyStream: TMemoryStream;
  C: String;
begin
  C := Edit1.Text;
  NMUDP1.RemoteHost := '127.0.0.1';
  NMUDP1.ReportLevel := Status_Basic;
  NMUDP1.RemotePort := 6668;
  MyStream := TMemoryStream.Create;
  try
    MyStream.Write(C[1], Length(C));
    NMUDP1.SendStream(MyStream);
  finally
    MyStream.Free;
  end;
end;
```

When Button1 is clicked, the text in Edit1 is written into the stream MyStream. The **SendStream** method is used to send the stream to the remote host, which is 127.0.0.1 (local host) in this instance. Note that MyStream is utilized within a **try**...**finally** loop to prevent a memory leak.

Insert the following code into NMUDP1's OnDataReceived event:

```
procedure TForm1.NMUDP1DataReceived(Sender: TComponent; NumberBytes: Integer; FromIP:
String; Port: Integer);
var
  C: String;
  MyStream: TMemoryStream;
begin
  MyStream := TMemoryStream.Create;
  try
    NMUDP1.ReadStream(MyStream);
    SetLength(C, NumberBytes);
    MyStream.Read(C[1], NumberBytes);
    Memo1.Lines.Add(C+': received '+IntToStr(NumberBytes)+' bytes from '+FromIP+' on port
'+IntToStr(Port));
  finally
    MyStream.Free;
  end;
end;
```

When data is received on the **LocalPort**, the **OnDataReceived** event is called. In this instance, a MemoryStream is created, and the **ReadStream** method is used to fill it with the incoming data. The incoming data is then read out of the stream into a string, and added to Memo1 along with the **NumberBytes** parameter, specifying the amount of incoming data, the **FromIP** parameter, specifying the computer that sent the data, and the **Port** property which specifies the port the data was sent from.

Note that MyStream is enclosed in a **try**...**finally** loop so it will be freed properly

Insert the following code into NMUDP1's OnDataSend event:

```
procedure TForm1.NMUDP1DataSend(Sender: TObject);
begin
  Memo2.Lines.Add('Data sent');
end;
```

When Button1 has been pressed and the data is sent successfully, the **OnDataSend** event is called, adding a message to Memo2 stating that the data has been sent.

Insert the following code into NMUDP1's OnStatus event:

```
procedure TForm1.NMUDP1Status(Sender: TComponent; status: String);
begin
  Memo2.Lines.Add(status);
end;
```

When a status message is generated, it is displayed in Memo2 when the **OnStatus** event is called by adding the **status** parameter.

Insert the following code into NMUDP1's OnInvalidHost event:

```
procedure TForm1.NMUDP1InvalidHost(var handled: Boolean);
var
  S: String;
begin
  S := NMUDP1.RemoteHost;
  if InputQuery('Invalid host', 'Specify valid hostname: ', S) then
  begin
    NMUDP1.RemoteHost := S;
    handled := TRUE;
  end;
end;
```

When the **OnInvalidHost** event is called, the InputQuery function is called, allowing the user to input a new host name to replace the invalid one previously entered. If the user clicks the Ok Button, the **handled** parameter is set to TRUE, and the component attempts it's action again with the new hostname. If the hostname is still invalid, or the cancel button was clicked in the dialog, an exception is raised.

Insert the following code into NMUDP1's OnStreamInvalid event:

```
procedure TForm1.NMUDP1StreamInvalid(var handled: Boolean; Stream: TStream);
begin
  ShowMessage('Invalid Stream: Stream contains no data');
end;
```

When the **OnStreamInvalid** event is called due to a stream containing no data, a message is displayed to the user stating that the stream is invalid because it contains no data. It would be possible to correct this error by adding data to the **Stream** parameter and changing the **handled** parameter to TRUE.

RemoteHost property

See also Example

Applies to
TNMUDP component

Declaration
```
property RemoteHost: string;
```

Description
The RemoteHost property specifies the dotted IP address or host name of the remote computer that is the target of the **SendBuffer** or **SendStream** methods.

Scope: Published
Accessability: Runtime, design-time

See also

RemotePort property

RemotePort property

See also Example

Applies to

TNMUDP component

Declaration

```
property RemotePort: integer;
```

Description

The **RemotePort** property specifies the port on the remote host to send data to using the **SendStream** and **SendBuffer** methods.

Scope: Published
Accessability: Runtime, design-time

See also

RemoteHost property
LocalPort property

ReportLevel property

See also Example

Applies to
TNMUDP component

Declaration
```
property ReportLevel: integer;
```

Description
The ReportLevel property specifies the level of detail in the status reporting given by the **OnStatus** event.

Default: Status_Informational

Range: You can use any of the following predefined contants to set the ReportLevel property:

Status_None
Status_Informational
Status_Basic
Status_Routines
Status_Debug
Status_Trace

Scope: Published
Accessability: Runtime, design-time

See also

OnStatus event

Create method

See also

Applies to

TNMUDP component

Declaration

```
constructor Create(AOwner: TComponent); override;
```

Description

The create method allocates memory and constructs a safely initialized instance of a component.
See TComponent.Create for details on inherited actions. TNMUDP also initializes WinSock when it is created.

See also

Destroy method

Destroy method

See also
Applies to
TNMUDP component

Declaration
```
destructor Destroy; override;
```

Description
You should not call the Destroy method in your application. Instead, call the Free method.
The Destroy method Disposes of the component and its owned components.See TComponent.Destroy for details on inherited actions. TNMUDP also cleans up WinSock when it is destroyed.

See also

Create method

ReadBuffer method

See also Example

Applies to

TNMUDP component

Declaration

```
procedure ReadBuffer(var Buff: array of char; var length: integer);
```

Description

The **ReadBuffer** method reads incoming UDP data into a Buffer.

Parameters:

The **Buff** parameter specifies the buffer to read the data into.
The **length** parameter is the size of the data to be read.

Notes:

The **Buff** parameter must be large enough to hold the size of the incoming data. If the buffer is to small, an access violation will occur.

See also

OnDataReceived event
ReadStream method

ReadStream method

See also Example

Applies to
TNMUDP component

Declaration
```
procedure ReadStream(DataStream: TStream);
```

Description
The ReadStream method is used to read UDP data being sent from a remote computer into a stream.

Parameters:
The **DataStream** parameter is any TStream or TStream descendent that has been initialized.

Notes:
If the stream passed as **DataStream** has not been initialized (it's Create method hasn't been called), an access violation will occur.

See also

OnDataReceived event
ReadBuffer method

SendBuffer method

See also Example

Applies to

TNMUDP component

Declaration

```
procedure SendBuffer(Buff: array of char; length: integer);
```

Description

The **SendBuffer** method is used for sending a buffer of data stored in an array of char to the remote host.

Parameters:

The **Buff** parameter is the array of characters that are to be sent to the remote host.
The **length** parameter specifies the length of the data in the Buff parameter.

Notes:

If the buffer passed as the **Buff** parameter contains no data, the **OnBufferInvalid** event is called.

See also

OnBufferInvalid event
SendStream method

SendStream method

See also Example

Applies to

TNMUDP component

Declaration

```
procedure SendStream(DataStream: TStream);
```

Description

The **SendStream** method is used to send a stream of data to the remote host.

Parameters:
The **DataStream** parameter is any TStream of TStream descendant that is to be sent to the remote host.

Notes:
If **DataStream** contains no data, the **OnStreamInvalid** event is called.

See also

OnStreamInvalid event
SendBuffer method

OnDataSend event

See also Example

Applies to

TNMUDP component

Declaration

```
property OnDataSend: TNotifyEvent;
```

Description

The OnDataSend event is called when UDP data has been sent successfully by either the **SendStream** or **SendBuffer** method.

Note:

The **OnDataSend** event does not guarantee that the data is received by the remote host. It only guarantees that it was sent. The UDP protocol does not acknowledge when data is received.

See also

SendBuffer method
SendStream method

OnInvalidHost event

See also Example

Applies to

TNMUDP component

Declaration

```
property OnInvalidHost: THandlerEvent;
```

Description

The **OnInvalidHost** event is called when the host specified by the **RemoteHost** property is invalid.

Event Parameters:

If the **handled** parameter is set to TRUE, the data is sent again. If it fails a second time due to the host being invalid, an exception is raised.
if the **handled** parameter is FALSE (the default), an exception is raised.

See also

RemoteHost property
SendBuffer method
SendStream method

TBuffInvalid type

Unit
NMUDP

Declaration
```
type
  TBuffInvalid = procedure(var handled: boolean; var Buff: array of char; var
length: integer) of object;
```

Description
The TBuffInvalid event type is used for the OnBufferInvalid event. TBuffInvalid is a modified THandler event type. If the **handled** parameter is set to FALSE (generally the default), the default actions will be taken by the component. If handled is set to TRUE, then the default component behavior is not taken.

THandlerEvent type

Unit
NMUDP

Declaration
```
type THandlerEvent = procedure(var handled: boolean) of object;
```

Description
The THandlerEvent event type is used for events that give the option of overriding the component's default behavior. Typically, if the **handled** parameter is FALSE (which is normally the default, see inidvidual event descriptions for details), teh component's default action is taken. If handled is TRUE, the component's default action is overridden.

TOnErrorEvent type

Unit
NMUDP

Declaration
```
type TOnErrorEvent = procedure(Sender: TComponent; errno: word; Errmsg:
string) of object;
```

Description
<<< Description of TOnErrorEvent type >>>

TOnReceive type

Unit
NMUDP

Declaration
```
type
  TOnReceive = procedure(Sender: TComponent; NumberBytes: Integer; FromIP:
string; Port: integer) of object;
```

Description
The TOnReceive event type is used for the handling of incoming data in the TNMUDP Component.

TOnStatus type

Unit
NMUDP

Declaration
```
type TOnStatus = procedure(Sender: TComponent; status: string) of object;
```

Description
The TOnStatus event type is used for events that report a status message.

TStreamInvalid type

Unit
NMUDP

Declaration
```
type
  TStreamInvalid = procedure(var handled: boolean; Stream: TStream) of
object;
```

Description
The TStreamInvalid event type is a modified THandler event type used for handling invalid streams. If the **handled** parameter is FALSE (the default value), the component's default actions are executed. If the handled parameter is TRUE, the default action taken by the component is ignored.

Heirarchy

TObject
|
TPersistent
|
TComponent

Legend

▶ Run-time only
▶ Read-Only
🔑 Published
🟨 Protected
🔑 Key item