# RealThinClient SDK – Demo 6 – Client to get a file from any Web Server.

In this demo we are building a client application that will be able to receive any file it request from a site, all of this with the Client components from **RealThinClient**.

**Basically we are going to:**

- Create a new Delphi project and add some controls
- Define events to process our request.
- Compile and run our project.

# 1. Start a new Project in Delphi

New Project

Set name of **Form1** to **fmMain**.


Form Name Property

# 3. Set the KeyPress property of your form to True

We will be catching the **Enter** key in any field so, when it is pressed, the system will start to search for the specified **Server Address** and **File Name**.


Key Preview Property

# 2. Put some controls in your form

Put a **Label** control

- Set **Caption** to **Server Address:**

Put a **TextEdit** control

- Set **Name** = **edServerAddress**
- Set **Text** = **www.realthinclient.net**

Put another **Label** control.

- Set **Caption** = **File:**

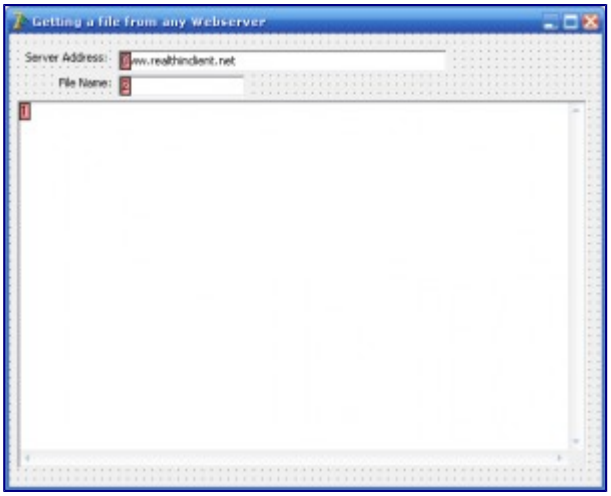
Put another **TextEdit** control

- Set **Name** = **edFileName**
- Set **Text** = **/**

Put one **Memo** control on your form

- Set **Name** = **mmResult**
- Set **Scrollbars** = **ssBoth**

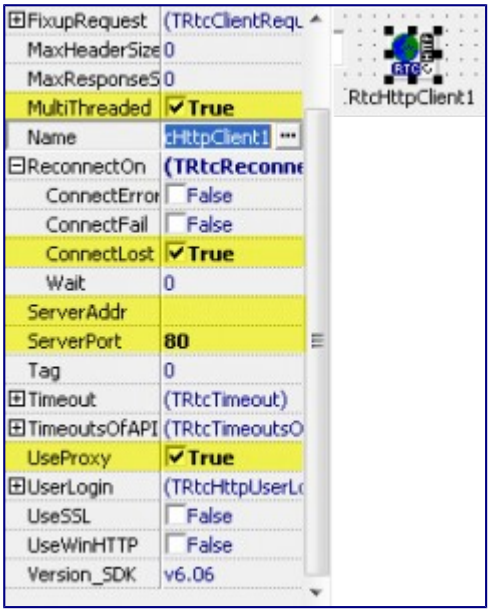
This is how our form should look with the controls placed in it:




Form Finished

# 3. Put some RTC controls in your form.

From the **RTC Client** tab, put one **RtcHttpClient** on your main form:

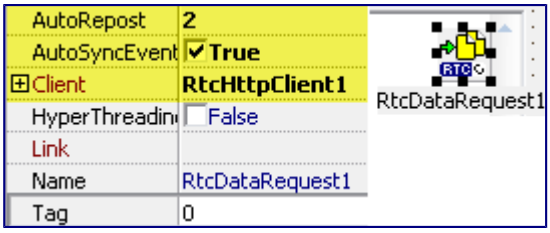



RtcHttpClient Control Properties

For **RtcHttpClient1**:

- Set **MultiThreaded** = **True**
- Set **ReconnectOn.ConnectLost** = **True**
- Set **ServerAddr** = **www.realthinclient.net**
- Set **ServerPort** = **80**
- Set **UseProxy** = **True**

From the **RTC Client** tab, put one **RtcDataRequest** on your form:




RtcDataRequest Control

For **RtcDataRequest1**:

- Set **Client** = **RtcHttpClient1**
- Set **AutoRepost** = **2**
- Set **AutoSyncEvents** = **True**

# 4. Define the OnBeginRequest event for RtcDataRequest control.

For **RtcDataRequest1**, define the **OnBeginRequest** event:

```
    procedure TfmMain.RtcDataRequest1BeginRequest(Sender:
1   TRtcConnection);
2   begin
3     with Sender as TRtcDataClient do
4     begin
5     // (1)
6     if Copy(Request.FileName, 1, 1) <> '/' then
7       Request.FileName := '/' + Trim(Request.FileName);
8     edFileName.Text := Request.FileName;
9     // (2)
10    if Request.Host = '' then
11      if Trim(edServerAddress.Text) <> '' then
12      begin
13        ServerAddr := edServerAddress.Text;
14        Request.Host := ServerAddr;
15      end;
16    Memo1.Lines.Clear;
17    Memo1.Lines.Add('Requesting "' + Request.FileName + ' From ' +
18  ServerAddr + '".');
19    // (3)
20    WriteHeader;
21    end;
    end;
```

What are we doing?

1. We check that our request starts with */*. If we make some change on the file name we reflect this change to the **edFileName** TextEdit control.
2. Then, we define the **HOST** header

3. After we have the **HOST** header defined, we send it out.

# 5. Define the OnDataReceived event for RtcDataRequest control

For **RtcDataRequest1**, define the **OnDataReceived** event:

```
procedure TfmMain.RtcDataRequest1DataReceived(Sender:
TRtcConnection);
begin
  with Sender as TRtcDataClient do
  begin
    if Response.Started then
    begin
      // (1)
      with mmResult do
      begin
        Lines.Add('   Status Code: ' +
IntToStr(Response.StatusCode));
        Lines.Add('   Status Text: ' + Response.StatusText);
        Lines.Add('   ALL Headers:');
        Lines.Add(Response.HeaderText);
        Lines.Add('Content Length:');
        Lines.Add(IntToStr(Response.ContentLength));
        Lines.Add('  Content Body:');
        Lines.Add('START >');
      end;
      // (2)
      mmResult.Text := mmResult.Text + Read;
      if Response.Done then
      begin
        // (3)
        mmResult.Lines.Add('> END');
        Request.Host := '';
        RtcHttpClient1.Disconnect;
      end;
    end;
  end;
end;
```

What are we doing?

1. We execute the **Header** info display on **Memo Control** one time per request, when we start receiving the response.
2. Depending on the content size, the line that show the content of the requested file could be executed several times.
3. When we have received all the content of the requested file, we print the **>END** tag and clear the Host name. Last, we disconnect our **RtcHttpClient** connection.

# 7. Define the OnKeyPress event for your form.

For **fmMain**, define the **OnKeyPress** event:

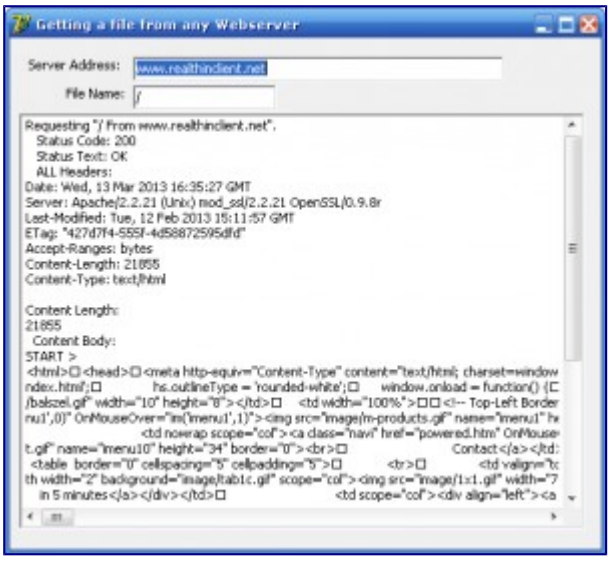
```
procedure TfmMain.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then
  begin
    //(1)
    edServerAddress.Text := Trim(edServerAddress.Text);
    if Copy(edServerAddress.Text, Length(edServerAddress.Text), 1)
= '/' then
      edServerAddress.Text := Copy(edServerAddress.Text, 1,
        Length(edServerAddress.Text) -1);
    RtcHttpClient1.ServerAddr := edServerAddress.Text;
    //(2)
    with RtcDataRequest1 do
    begin
      Request.Method := 'GET';
      Request.FileName := edFileName.Text;
      edServerAddress.SelectAll;
      edFileName.SelectAll;
      Post;
    end;
    //(3)
    Key := #0;
    //(4)
    RtcHttpClient1.Connect();
  end;
end;
```

What are we doing?

1. We check the **Server Address TextEdit** control so we make a few validations on it. We could use regular expressions to do this job, but this is out of the scope of this demo.
2. Configure our **TRtcDataRequest**. We are not making validations on the **FileName**, those are done whe the **Request** is started at the **OnBeginRequest** event.
3. No Beep.
4. Start the connection. We already set our **ServerAddr** property for our **RtcHttpClient** control, so now, we start our connection to that server.

# 20. Connect to the internet, then Compile and Run the project.

If every thing is ok, you should see your form ready to process your requests.




Project Running

In this example, content is received piece-by-piece and filled into the Memo field (mmResult) by adding the text Read to Memo's text and special care had to be taken to avoid clearing the memo in the middle of receiving.
You can access response header information in different ways:

- All headers as text using **Response.HeaderText**;
- Each header variable separately by using **Response['variable-name']**;
- Most header information specially prepared for easier access (**Response.ContentLength**, **Response.Query**, **Response.Cookie**, etc). The same rules apply to request info (**Request.HeaderText**, **Request['variable']**, **Request.ContentLength**, etc).

Even though **Response.ContentLength** could be undefined (zero) if Server only supports **HTTP/1.0** (server will end the response with a disconnect), you can always use the **Response.ContentIn** property to check how much of the response content has already arrived. This will increment for each package received, even if you do not call **Read**.

Other interesting events when receiving data are the **OnResponseDone** and **OnConnectLost** events. **OnResponseDone** event will be triggered after the last **OnDataReceived** event (after all data has been received), while the **OnConnectLost** event gets called in case your connection drops after executing the **OnBeginRequest** event, but before you received the complete response in the **OnDataReceived** event.

To store received data in a stream, you can open the stream from the **OnDataReceived** event when **Response.Started**, append (to the stream) everything you **Read** from the **OnDataReceived** event and close the stream from **OnResponseDone** and **OnConnectLost** events.