RealThinClient SDK Lesson 5 -Write your first Remote Function (Server and Client)

In this lesson, we'll be building a solution with a Server and a Client. This first lesson with a Client side made with **RealThinClient** Components is simple but shows the concept behind a client – server system that could be the base for a larger application. We'll be using **Remote Functions** to accomplish this task.

This Post has three sections.

1. **The Server.**
2. **The Client.**
3. **Make it work.**

1. The Server.

This is our first example on how to use remote functions. Basically, we are using:

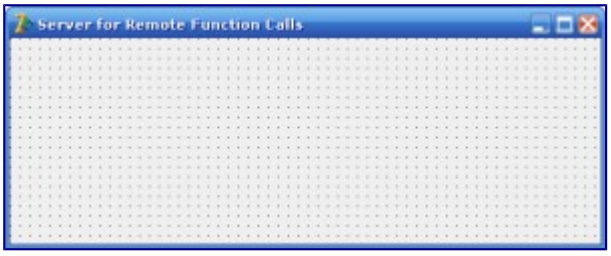
- **rtcServerModule**
- **rtcFunctionGroup** and
- **rtcFunction** components

With this three components we can write functions that can be called by our **RTC** clients.

Steps.

1.1. Open a new project.

We open a new project in our IDE.

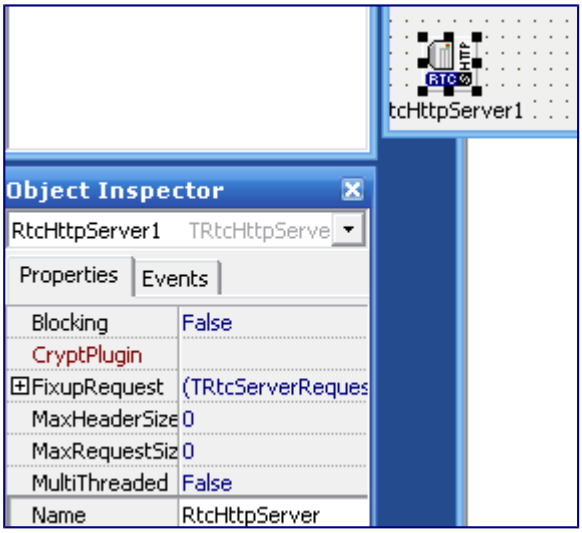


New Project

- Change the Caption property for Form1 to **Server for Remote Function Calls**.

1.2. Use a RtcHttpServer component.

Take a **RtcHttpServer** from your component's palette and drag it into your project's main form. Then, rename it to **RtcHttpServer**.



Add RtcHttpServer Component

Rename the **RtcHttpServer1** component to **RtcHttpServer**.

In the properties, set it's **Port** to **80**.

Set Port for RtcHttpServer to 80

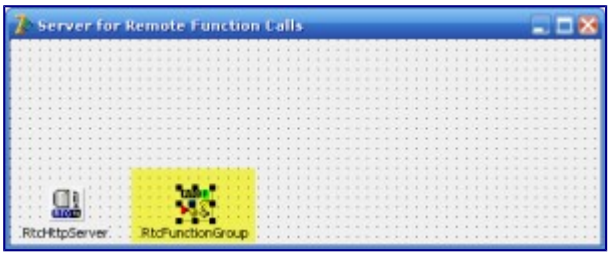
Now, define the **OnCreate** event for our main form in order to make our **RtcHttpServer** component start to listen for requests as soon as the application starts.

```
1procedure TForm1.OnCreate(Sender: TObject)
2begin
3  RtcHttpServer.Listen();
4end;
```

Now, we have our **RtcHttpServer** component configured and ready to take requests.

1.3. Use a RtcServerModule and a RtcFunction components.

5. from the **RTC Server** tab, put one **RtcFunctionGroup** on your form:



RtcFunctionGroup Component

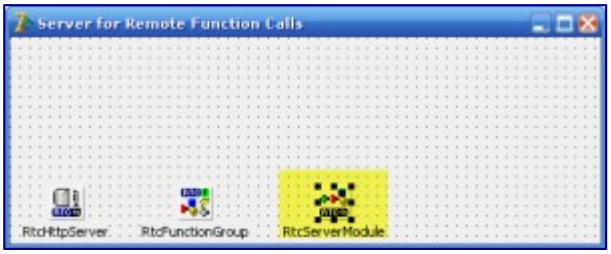
Rename it to **RtcFunctionGroup**

You will use one **FunctionGroup** for each Form or Module where you want to implement remote functions.

The **RtcFunctionGroup** component provides access to a group of remote functions.

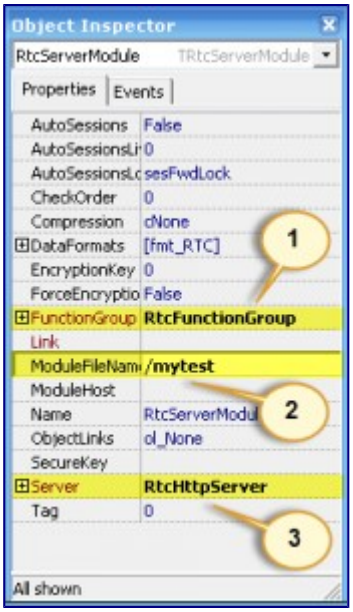
To implement remote functions, you will need at least one **RtcFunctionGroup** component and link one or more **RtcFunction** components to it. Function Group are used to make function calls as parameters to other function calls from the same group. It is primarily used by the **RtcServerModule** and **RtcClientModule** components to hold implementations for their remote functions.

Next, from the **RTC Server** tab, put one **RtcServerModule** on your form.



RtcServerModule Component

For **RtcServerModule**



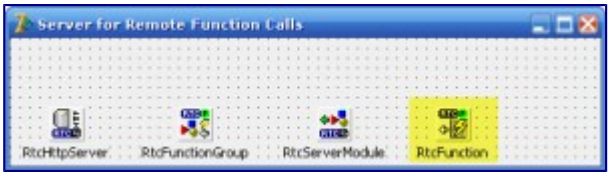
RtcServerModule Properties

1. Set **FunctionGroup** = **RtcFunctionGroup**
2. Set **ModuleFileName** = **/mytest**
3. Set **Server** = **RtcHttpServer**

ModuleFileName property is case-sensitive, so remember exactly what you use here, you will have to use exactly the same **ModuleFileName** in your **RtcClientModule** component for the **Client**.
You will use one **RtcFunction** component for each function you want to implement.

The **RtcServerModule** component accepts the request and uses a **TRtcFunctionGroup** component to execute received functions and prepares the result. If there are functions calls inside the objects received, those functions are executed and the resulting object will contain only data. The resulting object will be sent back to the client that made the request. In case of an exception the execution will be aborted and the object sent back to the client will be an exception message.

Now, from the **RTC Server** tab, put one **RtcFunction** on your form:



RtcFunction Component

RtcFunction Properties

- Set **FunctionGroup** = **RtcFunctionGroup**
- Set **FunctionName** = **Hello**

This **RtcFunction** is our remote function. To use it we need to define a function name, link it to a Function Group and define it's **OnExecute** event.

In case of an exception (which you can also raise inside your **OnExecute** event handler), the requesting client will get the exception message as a result.

You can combine multiple function calls in one request, or pass function calls as parameters to other function calls. It makes no difference to the functions you implement, because your function will always receive pure data, after all function calls (which the client might have defined as parameters) are executed.

And in case of serial function calls (more than one function called in one request), if one call ends up with an exception, the result for **that** call will be **rtc_Exception** (with the appropriate eror message), while any prior functions return their result and the execution of the request is aborted.

1.4. Code our RtcFunction's OnExecute event.

```
1procedure TForm1.HelloExecute(Sender: TRtcConnection;
2  Param: TRtcFunctionInfo; Result: TRtcValue);
3begin
4  Result.asString := 'Hello, ' + Param.asString['name']
5end;
```

1.5. Save, Compile and run our project to check that everything is working.

Save the Project, name it whatever you wish. Remember this is your **Server**, so make it meaningful. After you saved your project, compile and run it to check that everything is working here. Close it to create your **Client** application.

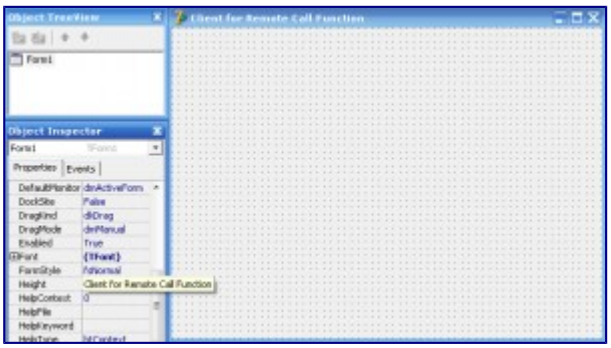
The Client.

This is the first client lesson for calling remote functions. You will learn in this lesson how to use **RtcClientModule** and **RtcResult** components to call remote functions served by a **RTC** server.

Steps.

2.1. Create a new Project in your IDE

Create a new Project in the IDE and rename the main form to fmMain

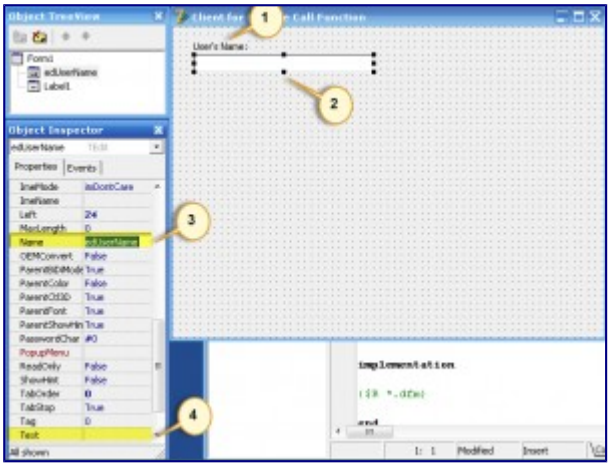


Client New Project

2.2. Put a few controls in your Client’s main form.

For our client’s application, we are going to need three components for user interaction:

- A Label
- A TextEdit
- A Memo



Add TLabel and TEdit components

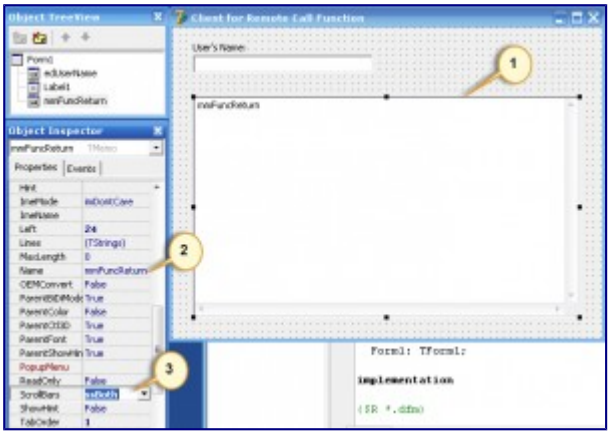
Drag those three components to your main form and
With Label1

- Rename it to **lbUserName**
- Set it's Caption property to **User Name:**

TEdit Properties

With Edit1

- Rename it to **edUserName**
- Set it's Text property to **User's Name**



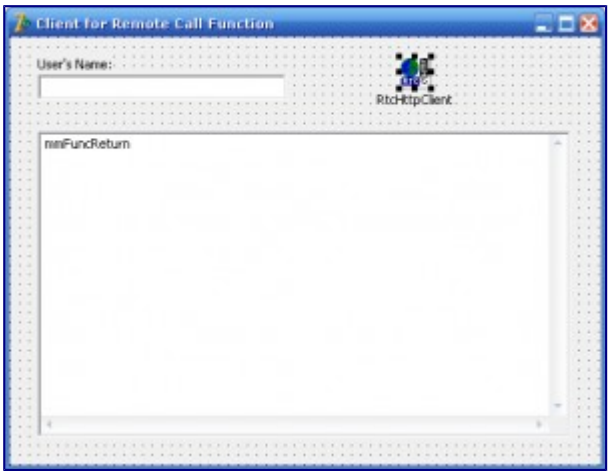
Add a TMemo Control

With Memo1

- Rename it to **mmMemo**
- Set it's **ScrollBars** property to **ssBoth**
- Clean it's **Strings** property

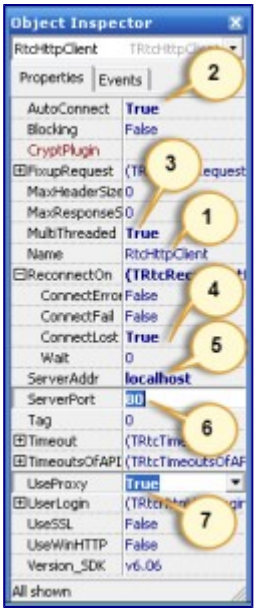
2.3. Use a RtcHttpClient component

We are creating a Client, we already have our Server application, so now we need a **RtcHttpClient** component to comunicate with our server. Drag one **RtcHttpClient** component to the main form.



Add RtcHttpClient Control

With RtcHttpClient1 component



RtcHttpClient Properties

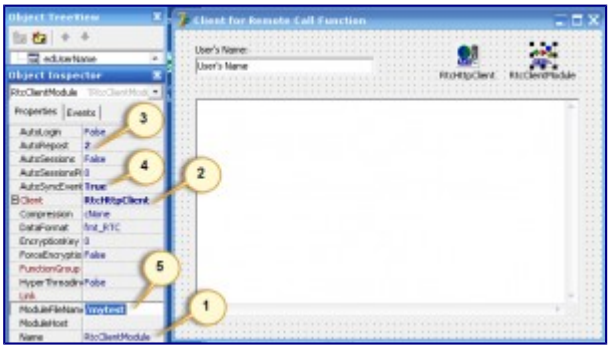
1. Rename it to **RtcHttpClient**
2. Set it's **AutoConnect** property to **True**
3. Set it's **MultiThreaded** property to **True**
4. Set it's **ReconnectOn**.**ConnectLost** property to **True**
5. Set it's **ServerAddr** propety to **localhost**
6. Set it's **ServerPort** property to **80**
7. Set it's **UseProxy** property to **True**

The **RtcHttpClient** component is used for TCP/IP communication using HTTP requests. Received data will be processed by **RtcHttpClient** to gather **Request** information and make it easily accessible through the **Request** property. The same way, your response will be packed into a HTTP result header and sent out as a valid HTTP result, readable by any Web Browser.

RtcHttpClient also makes sure that you receive requests one by one and get the chance to answer them one-by-one, even if the client side sends all the requests at once (as one big request list), so you can relax and process all incoming requests, without worrying about overlapping your responses for different requests.

2.4. Use a RtcClientModule component

Drag one **RtcClientModule** component to the main form.



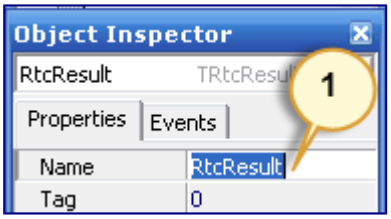
RtcClientModule Properties

1. Rename it to **RtcClientModule**
2. Set it's **Client** property to **RtcHttpClient**
3. Set it's **AutoRepost** property to **2**
4. Set it's **AutoSyncEvents** to **True**
5. Set it's **ModuleFileName** to **/mytest**

RtcClientModule is used to prepare remote function calls, post them to the **Server**, accept **Server's** response and call local event handlers with the result received for each call.

2.5. Use a RtcResult component

Drag one **RtcResult** component to the main form.

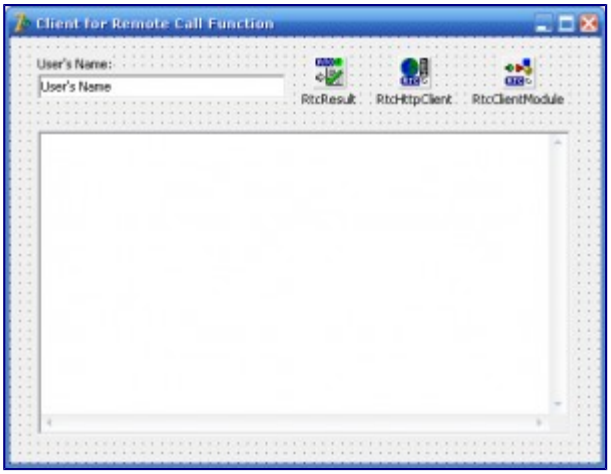


RtcResult Properties

1. Rename it to **RtcResult**

This component will be used to process the result returned from the remote function call, to do this, we must implement it's **OnReturn** event.

Our form should look like this



Form With Controls

2.6. Define the OnReturn event for our RtcResult component

```
procedure TfmMain.RtcResultReturn(Sender: TRtcConnection; Data,
  Result: TRtcValue);
begin
  mmMemo.Lines.Add(Result.asString);
end;
```

2.7. Define the OnKeyPress event for our edUserName component

```
1
2
3
4
5 procedure TfmMain.edUserNameKeyPress(Sender: TObject; var Key:
6 Char);
7 begin
8   if Key=#13 then
9   begin
1     edUserName.SelectAll;
0     with RtcClientModule do
1     begin
1       // (1)
1       with Data.newFunction('Hello') do
2       begin
1         // (2)
3         asString['name'] := edUserName.Text;
1       end;
4       // (3)
1       Call(RtcResult);
5     end;
1   end;
6 end;
1
7
1
8
```

The **Enter** key will trigger our Remote Function call. To do this, we are catching the **Enter** key code (#13) and then:

1. Prepare a new function call.
2. Set function's call parameters and
3. Call the remote function.

3. Make it Work

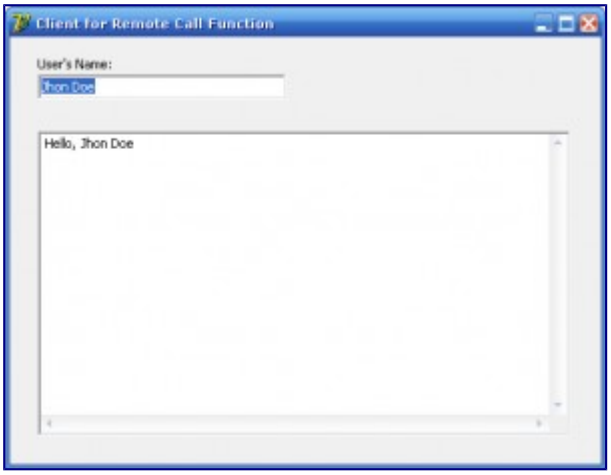
3.1. Start the Server

You must have an **exe** that was created when you compiled your **Server's** project. So, now run it, it'll show you the project's form on the screen.

Server Running

3.2. Start the Client and do your tests

Now, compile and run your client's application, when you enter some text in the Edit field and press the **Enter** key, you will receive a response in your **mmMemo** component.



Client Running