

XML-RPC Client 2

by Thom McGrath

Introduction

The original classes allowed use of the XML-RPC protocol from within REALbasic. While technically functional, they lacked in a couple key areas. Most notably, they required an experienced REALbasic user who knew typecasting well. There was no demo project, they provided minimal feedback during transactions, and did not support secure servers.

That's all be changed. The new version has been completely rewritten from the ground-up. Now, rather than 8 value classes, everything is done using Variants. That means you can use native data types, and responses are significantly easier to use.

The client is built on HTTPSecureSocket, and automatically supports SSL as detected by the server URL. Because of this, the client now provides progress events to provide developers real time feedback.

Requirements

This new project makes extensive use of the new Pairs class introduced in REALbasic 2008r2. Thus, you'll need that version or newer. Because the client is built on HTTPSecureSocket, you'll also need REALbasic Professional. It is possible (and pretty simple) to change the super to HTTPSocket, and change the code within. Feel free to do so, but I can't help you much with it.

Usage

You have two options for using the Client class. Which method you use depends entirely on your project requirements. The first option is to create a class with a super of XMLRPC.Client. The second option is to drag the XMLRPC.Client class onto a window. In either case, you'll use the events provided by the Client class to interact with the data. After that, set the RPCServer property to the complete url of the XML-RPC server, "http://" included. (This is in contrast to the previous version where you would leave the scheme off.) There is an RPCPort option as well, which will automatically set itself. Changes made to this variable will override the default. To restore the behavior, change RPCPort to 0. In nearly all cases, you'll never need to worry about the RPCPort.

Now, when you're ready to send a message, create a new XMLRPC.Message. You can pass the method name into the constructor, or use the MethodName property. Now, add the parameters, in order. There is a convenience method AddParam which takes a variant. AddParam will attempt to guess the correct parameter type, and usually, it will be correct. It will not be able to detect binary/base64 data correctly, however, and will set the parameter type to a string. To make certain your parameters are added to the message properly, there are a series of AddParam_* methods: Array, Binary, Boolean, Date, Dictionary, Double, Integer, and String.

Important note about arrays: Arrays passed to the Message class, even if inside a dictionary, must be declared as a variant, otherwise you will get a TypeMismatchException.

When you are ready to send your message to the server, you have two options: Client.SendMessage or Client.SendSynchronousMessage. The former has no return value, and the ResponseReceived event will fire instead as well as the other status events. The latter returns the response as a variant, and no status events except the Error event will fire.

Handling a response is simple. In most cases, you can use it like you would use any variable. If the response is an array or dictionary, you'll need to take special care. The demo project should make it easy to determine how to do this.

Errors

There are a number of new errors codes in this version.

XMLRPC.Client.kError_BadXML = 500

The server returned invalid or non-xml data.

XMLRPC.Client.kError_NoResponse = 501

The server was found, but returned no data.

XMLRPC.Client.kError_HTTPError = 502

The server replied, but something went wrong. The message will contain more information.
XMLRPC.Client.kError_ServerNotFound = 503
The server could not be found
XMLRPC.Client.kError_ConnectionLost = 504
The server terminated the connection
XMLRPC.Client.kError_SocketError = 505
There was an error with SocketCore. The message will contain more information.
XMLRPC.Client.kError_UnknownMethod = 506
The server received and processed the message, but it does not have a procedure for the requested method.

Contact Information

You can reach me via e-mail at zero@thezaz.com

License

You may use the classes as you need, they're open source. You may only redistribute if the classes and documentation are unmodified from their original form.

If you make changes, e-mail them to me. If they work out well, I'll roll them into a new release and give credit where credit is due. But please, don't start distributing modified versions - that's just a mess.