

rpc_plugin

rpc 远程调用插件，一条语句创建 C/S 架构应用。

安装

动态库文件：tsl\plugin\rpc_plugin.dll

文档：https://git.mytsl.cn/tinysoft/rpc_plugin.git

使用指南

服务端：

1、`rpc_server(host:string, threadcnt:integer,token:string, [UserVerifyFunc:string]);`

功能：启动 rpc 服务器。

参数说明：

`host` 字符串，服务器地址，如：“0.0.0.0:1234”。

`threadcnt` 整数，服务器开启的线程池数。

`token` 字符串，通讯验证码，通讯加密传输。

`[UserVerifyFunc]` 字符串，可选参数，用户验证函数名，用户可以编写自定义的验证方法，`rpc_login` 时会被调用，函数定义 `Function UserVerifyFunc(UserAuthCode:String, LoginTime:String)`，返回 1 或者非空字符串（`AuthCode`，必须与 `rpc_login` 输入一致），允许登录。

返回值：`[err, val]`，`err=0` 启动成功，否则返回错误码。

DEMO：

```
[err, val] := rpc_server("0.0.0.0:1234", 5, "my-token_value");
```

用户验证方式：

- ① **缺省验证**：`UserVerifyFunc` 为空，采用系统缺省验证方式；
- ② **证书验证**：需要用户扩展 `UserVerifyFunc` 函数，客户端用私钥加密信息，通过 `UserAuthCode` 传到服务端 `UserVerifyFunc` 函数，`UserVerifyFunc` 函数用服务器端公钥解密信息，返回 1 或非空字符串（`AuthCode` 与 `rpc_login` 中参数一致）表示校验通过；
- ③ **双因素验证**：`UserAuthCode` 信息包含动态挑战码的话，用户可以扩展支持双因素

认证。

- ④ **证书加密+双因素验证**：验证方式 2 中，加密信息包含动态挑战码的话，用户可以扩展支持证书加密+双因素认证。

2、`rpc_register_func(func1:string, func2:string...);`

功能：注册服务端能被远程调用的函数名。不注册的话，默认所有的 TSL 函数都能被远程调用，必须在 `rpc_server` 函数之前调用。

参数说明：

Func1、func2、... 字符串，TSL 函数名。

返回值：无。

DEMO：

```
rpc_register_func("func1", "func2",...);
```

3、`rpc_register_number(fno1:string, func1:string...);`

功能：注册服务端能被远程调用的功能号，注册后客户端必须采用功能号调用服务端应用。不注册的话，默认所有的 TSL 函数都能被远程调用，必须在 `rpc_server` 函数之前调用，`rpc_register_number` 和 `rpc_register_func` 两种注册方法二选一。

参数说明：

fno1 字符串，TSL 函数对应的功能号。

Func1 字符串，TSL 函数名。

... fno, func 参数成对出现。

返回值：无。

DEMO：

```
rpc_register_number("100", "now",...);
```

//这样客户端就不能通过函数名调用服务器端程序了，必须这样：

```
rpc_call(h, 10000, "100"); //通过功能号访问服务器时间
```

访问服务端应用：

设置方式	调用服务端方式	是否能访问全部函数
不设置	函数名	全部
<code>rpc_register_func</code>	函数名	注册的函数
<code>rpc_register_number</code>	功能号	通过功能号注册的函数

日志:

服务器端日志记录在 `tsl\log\yyyy-mm-dd.log` 中

```
pid:49544 tid:46792 tm:08:49:32 ws:961133120 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:180609024->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:41312 tm:08:50:03 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:46792 tm:08:50:13 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:46792 tm:08:50:50 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:46792 tm:08:51:19 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:51752 tm:08:51:37 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:51752 tm:08:51:41 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:51752 tm:08:51:48 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:51752 tm:08:51:54 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:46792 tm:08:51:59 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:51752 tm:08:52:04 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
pid:49544 tid:46792 tm:08:52:08 ws:96116736 pws:348168192 pppu:653112 ppu:613400 pnppu:5338336 nppu:93240 pfu:178941952->[RPC] Rpc_exec: return tsdb_version(); ->OK.
```

客户端:

4、`rpc_login(host:string, UserAuthCode:string, token:string, [threadcnt:integer], [AuthCode:string]);`

功能: 登录 rpc 服务器。

参数说明:

`host` 字符串, 服务器地址, 如: "192.168.1.10:1234"。

`UserAuthCode`, 字符串, 用户自定义验证码, 用户可以扩展自定义验证算法, 参考 `rpc_server` 函数说明。

`token` 字符串, 通讯验证码 (要求与 `rpc_server` 一致), 通讯加密传输。

`threadcnt` 整数, 可选参数, 默认 3, 网络线程池数。

`AuthCode` 字符串, 可选参数, 通讯密钥与该值相关, 如果设置该值, 必须在 `UserAuthCode` 中存储该值的密文, 在服务器端 `UserVerifyFunc` 函数能解析出并返回该值, 否则客户端和服务端通讯密钥不一致。

返回值: `[err, handle]`, `err=0` 登录成功, 否则返回错误码, `handle` 为连接句柄。

DEMO:

```
[err, h] := rpc_login("192.168.1.10:1234", "user", "my-token_value");
```

5、`rpc_logout(handle:integer);`

功能: 登出 rpc 服务器。

参数说明:

`Handle` 整数, `rpc_login` 登录成功返回的连接句柄。

返回值: `[err, val]`, `err=0` 成功登出, 否则返回错误码。

DEMO:

```
[err, h] := rpc_logout(h);
```

6、rpc_ping(handle:integer, ping:integer);

功能：ping RPC 服务器，测试网络连接状况。

参数说明：

Handle 整数，rpc_login 登录成功返回的连接句柄。

ping 整数，传递给 RPC 服务器的值。

返回值：[pong, val]，成功 pong=ping。

DEMO：

```
[pong, v] := rpc_ping(h, 666); //pong=666
```

7、rpc_call(handle:integer, timeout:integer, func:string, [arg1:Any],[arg2:Any]...);

功能：RPC 远程调用服务端 TSL 函数。

参数说明：

Handle 整数，rpc_login 登录成功返回的连接句柄。

timeout 整数，超时毫秒数（超时返回错误）。

Func 字符串，远程服务器 TSL 函数名。

Arg1, Arg2... 任意类型，远程函数调用参数。

返回值：[err, retval]，err=0，远程调用成功，retval 为远程服务器函数执行结果。

DEMO：

```
[err, t] := rpc_call(h, 10000, "now"); //返回服务器时间。
```

```
echo "servertime: ",datetimeostr(t); //servertime: 2022-07-27 10:29:10
```

```
[err, retval] := rpc_call(h, 10000, "rand", 10, 10); //返回服务器端生成的 10*10 随机矩阵。
```

```
[err, val] := rpc_call(h, 10000, "myfunction", arg1, arg2, arg3, arg4); //调用服务端自定义函数
```