**RRQMSocket使用手册**

**适用版本：2.0.7**

目录

[**一、程序集描述** 2](#_Toc59525210)

[**二、程序集功能概述** 2](#_Toc59525211)

[**2.1文件传输类概述** 2](#_Toc59525212)

[**三、程序集具体使用说明** 3](#_Toc59525213)

[**3.1常用类继承图** 3](#_Toc59525214)

[**3.2服务器类（TcpService）** 3](#_Toc59525215)

[**3.2.1 创建自定义服务器类（SimpleTcpService）** 4](#_Toc59525216)

[**3.2.2 文件服务器类（FileService）** 5](#_Toc59525217)

[**3.2.3 RPC服务器类（RPCService）** 7](#_Toc59525218)

[**3.2.4 RPC基于RRQMTcp协议的服务器类（RRQMRPCService）** 7](#_Toc59525219)

[**3.3客户端类（TcpClient）** 9](#_Toc59525220)

[**3.3.1 创建自定义客户端类（SimpleTcpClient）** 10](#_Toc59525221)

[**3.3.2 创建RRQMTcp协议的客户端类（RRQMTcpClient）** 10](#_Toc59525222)

[**3.3.3 文件客户端类（FileClient）** 11](#_Toc59525223)

[**3.3.4 RPC客户端类（RPCClient）** 12](#_Toc59525224)

[**3.3.5 文件传输类使用注意事项** 15](#_Toc59525225)

[**四、使用其他功能** 15](#_Toc59525226)

[**4.1设置最大上传和下载速度** 15](#_Toc59525227)

[**4.2加密和解密** 15](#_Toc59525228)

[**4.3序列化和反序列化** 15](#_Toc59525229)

[**五、致谢** 15](#_Toc59525230)

**一、程序集描述**

**该程序集由作者本人独自开发，设计思路和基本代码也是独自开发，一小部分成熟代码来源于网络，属于开源代码，不会有版权争议。程序集本身限定功能可供大家免费学习，但不能用于商业，产生商业利益，若有商业需求，请联系作者QQ505554090。特此供知。**

**二、程序集功能概述**

**2.1文件传输类概述**

文件传输类是基于TCP连接，整合功能性的类，其主要功能如下：

**（1）上传文件：**

* 上传文件
* 断点续传
* 设置最大上传速度
* 获取实时上传速度
* 获取上传进度
* 获取上传文件信息
* 暂停上传
* 恢复上传
* 终止上传

**（2）下载文件：**

* 下载文件
* 断点续传
* 队列式下载文件
* 设置最大下载速度
* 获取实时下载速度
* 获取下载进度
* 获取下载文件信息
* 暂停下载
* 恢复下载
* 终止当前下载
* 终止所有下载

**（3）传递数据：**

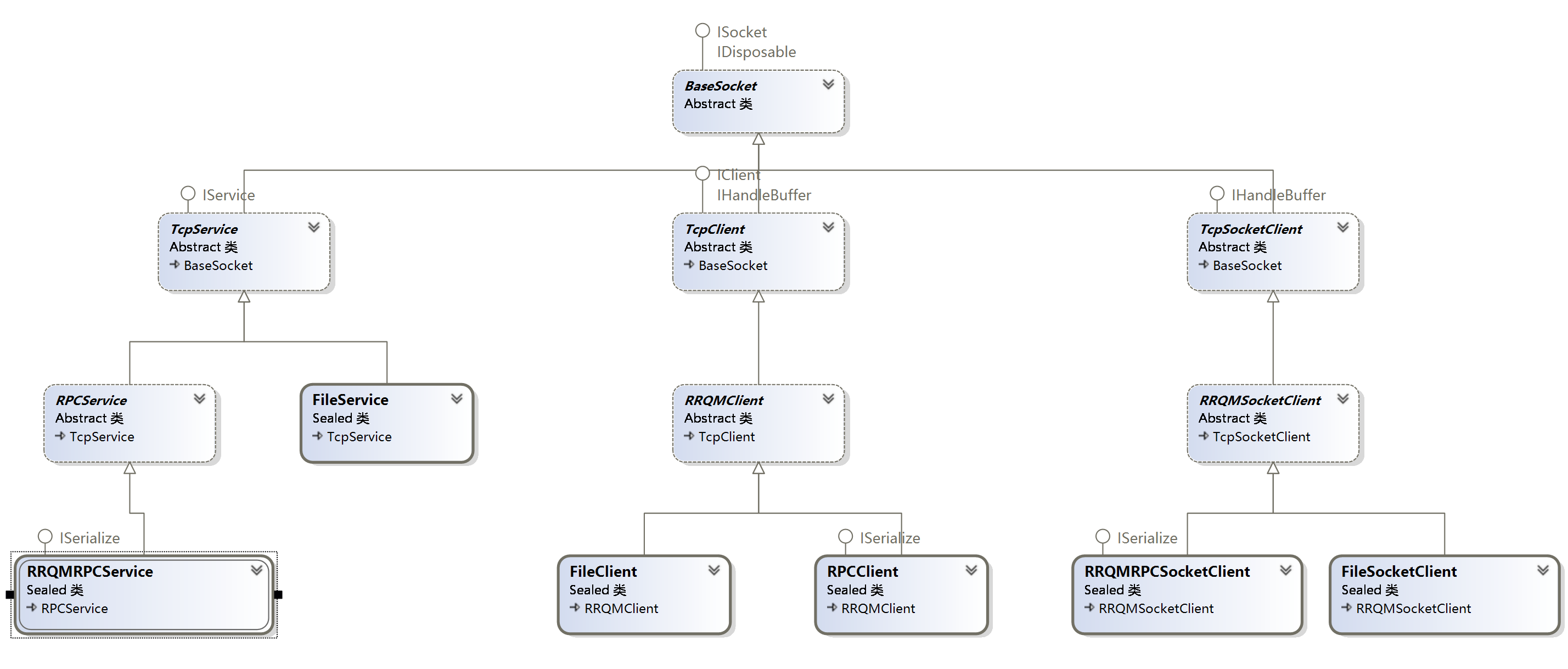
* 传递任意可序列化（xml、二进制、json）的数据对象
* 传递任意大小的字符串

**（4）参数设置：**

* 设置客户端连接的用户名和密码

**三、程序集具体使用说明**

**3.1常用类继承图**



**3.2服务器类（TcpService）**

TcpService是所有基于TCP协议类的基类，也是在服务器端创建服务器的对象类，但该类是抽象类，必须通过继承才能创建其具体对象，原因是在该类中，只实现了服务器的管理功能，但未实现其数据接收和处理功能，所以必须倚仗服务器辅助类才能实现其实际功能。其简单介绍如下：

a.可设置属性

**LicenceKey**：使用许可证，专业版必要参数。

**IsCheckClientAlive**：是否自动检验客户端活性。

b.只读属性

**IP**:绑定服务器所用到的IP地址。

**Port**：绑定服务器所用到的端口号 。

**IsBand**：判断服务器绑定是否成功。

**LicenceDate**：激活使用日期，购买时由密钥决定。

**MaxConnection**：同时可连接的最多客户端数量，购买时由密钥决定。

**UserMessage**：服务器授权人信息，购买时由密钥决定。

**3.2.1 创建自定义服务器类（SimpleTcpService）**

1. **public** **class** SimpleTcpService : TcpService
2. {
3. #region Methods
5. /// <summary>
6. ///重写TcpSocketClient的获取方式，类似于SimpleTcpSocketClient类，该类继承于抽象类TcpSocketClient
7. ///该方法的主要作用是生成用于和客户端通信的辅助类
8. /// </summary>
9. /// <returns></returns>
10. **protected** **override** TcpSocketClient CreatSocketCliect()
11. {
12. SimpleTcpSocketClient socketClient = **new** SimpleTcpSocketClient();
13. **return** socketClient;
14. }
16. #endregion Methods
17. }
18. /// <summary>
19. /// 该类是简单TCP的辅助类
20. /// </summary>
21. **public** **class** SimpleTcpSocketClient : TcpSocketClient
22. {
23. **int** a = 0;
24. #region Methods
26. /// <summary>
27. /// 该方法的主要作用是处理接收到的TCP流数据
28. /// 该数据与发送方不完全一致，可能会发生粘包，分包等情况。
29. /// </summary>
30. /// <param name="buffer"></param>
31. /// <param name="r"></param>
32. **public** **override** **void** HandleBuffer(**byte**[] buffer, **int** r)
33. {
34. **if** (++a%100==0)
35. {
36. Console.WriteLine(a);
37. }
38. **this**.Send(buffer,0,r);//通过默认Send方法，把收到的数据原数返回，该数据亦不完全和Client收到的数据一致，可能会发生粘包，分包等情况。
39. }
41. #endregion Methods
42. }

**3.2.2 文件服务器类（FileService）**

**（1）主要属性及说明：**

服务器类

a.可设置属性

**LicenceKey**：使用许可证，专业版必要参数。

**BreakpointResume**：是否支持断点续传

**IsFsatTransmission**：是否支持快速上传

b.只读属性

**IP**:绑定服务器所用到的IP地址。

**Port**：绑定服务器所用到的端口号 。

**IsBand**：判断服务器绑定是否成功。

**LicenceDate**：激活使用日期，购买时由密钥决定。

**MaxConnection**：同时可连接的最多客户端数量，购买时由密钥决定。

**UserMessage**：服务器授权人信息，购买时由密钥决定。

**DownloadSpeed**：总下载速度。

**UploadSpeed**：总上传速度。

**（2）主要事件及说明：**

|  |  |
| --- | --- |
| 事件名 | 事件触发及描述 |
| ClientConnected | 有用户成功连接以后触发 |
| ClientDisconnected | 由用户主动断开连接，或者由服务器遍历时检测出连接已中断后触发 |
| BeforeReceiveFile | 接收到客户端上传文件请求之后触发 |
| BeforeSendFile | 当客户端请求下载文件，且已经判断请求路径合法，即将要给客户端回应下载请求时触发 |
| SendFileFinished | 给客户端发送完文件时触发 |
| ReceiveFileFinished | 接收完客户端上传的文件时触发 |

**（3）主要方法及说明：**

|  |  |  |
| --- | --- | --- |
| 方法名 | 方法功能及描述 | 参数类型及说明 |
| Bind | 绑定服务器 | BindSetting.IP：IP地址  BindSetting.Port：端口号  BindSetting. MultithreadThreadCount：处理数据的多线程数量 |
| Dispose | 释放资源 | 无 |

**（4）使用：**

1. **private** **void** CreatService()
2. {
3. fileService = **new** FileService();
4. fileService.IsFsatUpload = **true**;//快速传输
5. fileService.BreakpointResume = **true**;//断点续传
6. BindSetting setting = **new** BindSetting();
7. setting.IP = "127.0.0.1";
8. setting.Port =7788;
9. setting.MultithreadThreadCount = 2;//多线程，线程数量为2
10. fileService.Bind(setting);
11. fileService.ClientConnected += FileService\_ClientConnected;
12. fileService.ClientDisconnected += FileService\_ClientDisconnected;
13. fileService.BeforeReceiveFile += FileService\_BeforeReceiveFile;
14. fileService.BeforeSendFile += FileService\_BeforeSendFile;
15. fileService.ReceiveSystemMes += FileService\_ReceiveSystemMes;
16. fileService.ReceiveFileFinished += FileService\_ReceiveFileFinished;
17. fileService.SendFileFinished += FileService\_SendFileFinished;
18. }

**3.2.3 RPC服务器类（RPCService）**

**RPCService是RPC服务器的基类，在TcpService基础上没有过多的增强，只定义了开启RPC的方法，其使用比较简单。**

**3.2.4 RPC基于RRQMTcp协议的服务器类（RRQMRPCService）**

**RRQMRPCService是本程序集实现的具体的一个RPC服务器，其功能可以实现全部的远程过程调用的功能，其包括代理代码生成，错误异常反馈，调用异常反馈，out及ref关键字的使用，泛型类型的使用，泛型类型的out及ref的使用等。具体说明如下：**

**（1）主要属性及说明：**

a.可设置属性

**LicenceKey**：使用许可证，专业版必要参数。

**SerializeType**：序列化方式，有二进制，xml和自定义。

**SerializeConverter**：当序列化类型设置为自定义的时候，则通过该序列化器实现

b.只读属性

**IP**:绑定服务器所用到的IP地址。

**Port**：绑定服务器所用到的端口号 。

**IsBand**：判断服务器绑定是否成功。

**LicenceDate**：激活使用日期，购买时由密钥决定。

**MaxConnection**：同时可连接的最多客户端数量，购买时由密钥决定。

**UserMessage**：服务器授权人信息，购买时由密钥决定。

**RPCServerMode**：RPC服务器模式，一般模式不能通过

**RPCServerVersion**：总上传速度。

**（2）主要事件及说明：**

|  |  |
| --- | --- |
| 事件名 | 事件触发及描述 |
| ClientConnected | 有用户成功连接以后触发 |
| ClientDisconnected | 由用户主动断开连接，或者由服务器遍历时检测出连接已中断后触发 |

**（3）主要方法及说明：**

|  |  |  |
| --- | --- | --- |
| 方法名 | 方法功能及描述 | 参数类型及说明 |
| Bind | 绑定服务器 | BindSetting.IP：IP地址  BindSetting.Port：端口号  BindSetting. MultithreadThreadCount：处理数据的多线程数量 |
| OpenRPCServer | 打开RPC服务器 | RPCServerSetting.RPCServerMode：服务器模式  RPCServerSetting.NameSpace：高级RPC的代理程序集命名空间 |
| Dispose | 释放资源 | 无 |

**（4）使用：**

1. RRQMRPCService rpcService = **new** RRQMRPCService();
3. BindSetting bindSetting = **new** BindSetting();
4. bindSetting.IP = "127.0.0.1";
5. bindSetting.Port = 7789;
6. bindSetting.MultithreadThreadCount =1;
7. rpcService.Bind(bindSetting);
9. RPCServerSetting setting = **new** RPCServerSetting();
10. setting.RPCServerMode = RPCServerMode.Senior;
11. setting.NameSpace = "RRQMTest";
12. rpcService.OpenRPCServer(setting);
14. rpcService.SerializeType = SerializeType.Define;
15. //序列化方式选择自定义的时候，必须对SerializeConverter属性赋值
16. //类似于DefineSerialize类，该类继承于SerializeConverter抽象类，并强制重写序列化和反序列化方法。
17. //序列化方式应该让服务器端和客户端保持一致，当然不局限于同一个类，但是也要能对数据进行正确的序列化和反序列化。
18. rpcService.SerializeConverter = **new** DefineSerialize();
19. /// <summary>
20. /// 自定义序列化类
21. /// </summary>
22. **public** **class** DefineSerialize : SerializeConverter
23. {
24. /// <summary>
25. /// 反序列化
26. /// </summary>
27. /// <param name="buffer"></param>
28. /// <returns></returns>
29. **public** **override** RPCMethod Deserialize(**byte**[] buffer)
30. {
31. //用RRQMSocket自带的反序列化方法模拟
32. //使用者可以根据自身需要，选择json等反序列化。
33. **return** SerializeConvert.DeserializeWithBinary<RPCMethod>(buffer, 0, buffer.Length);
34. }
36. /// <summary>
37. /// 序列化
38. /// </summary>
39. /// <param name="waitResult"></param>
40. /// <returns></returns>
41. **public** **override** **byte**[] Serialize(RPCMethod waitResult)
42. {
43. //用RRQMSocket自带的序列化方法模拟
44. //使用者可以根据自身需要，选择json等序列化。
45. **return** SerializeConvert.SerializeToBinary(waitResult);
46. }
47. }

**3.3客户端类（TcpClient）**

TcpClient是所有基于TCP协议客户端类的基类，必须通过继承才能创建其具体对象，原因是在该类中，只实现了简单的数据发送和接收，但未实现其数据的处理功能，所以必须在继承类中实现其实际功能。其简单介绍如下：

**（1）主要属性及说明：**

1）可设置属性

2）只读属性

**Connected**：连接状态。

**IP**：连接IP地址。

**Port**：连接端口号。

**Name**：连接到服务器的唯一标识（IP及端口号）。

**3.3.1 创建自定义客户端类（SimpleTcpClient）**

1. /\*
2. 若汝棋茗
3. \*/
5. /// <summary>
6. /// 该类是基于Socket的简单TCP的客户端类
7. /// 通过该类发送的字节流不完全和接收方保持一致
8. /// 可能会发生分包、粘包等情况
9. /// </summary>
10. **public** **class** SimpleTcpClient : TcpClient
11. {
12. #region Methods
14. /// <summary>
15. /// 该方法主要作用是对服务器发送来的字节流进行处理
16. /// 该数据与发送方不完全一致，可能会发生粘包，分包等情况。
17. /// 当然也如果服务器没有返回数据的话，也不会调用该方法。
18. /// </summary>
19. /// <param name="buffer"></param>
20. /// <param name="r">接收长度</param>
21. **public** **override** **void** HandleBuffer(**byte**[] buffer, **int** r)
22. {
23. **string** mes = Encoding.UTF8.GetString(buffer,0,r);
24. Console.WriteLine($"已接收到信息：{mes}");
25. }
27. #endregion Methods
28. }

**3.3.2 创建RRQMTcp协议的客户端类（RRQMTcpClient）**

1. /// <summary>
2. /// 该类是基于TcpClient的TCP客户端类
3. /// 通过该类发送的字节流完全和接收方保持一致
4. /// </summary>
5. **public** **class** RRQMTcpClient : RRQMClient
6. {
7. **int** a;
8. **protected** **override** **void** RRQMTcpHandleBuffer(**byte**[] buffer)
9. {
10. **string** mes = Encoding.UTF8.GetString(buffer);
12. **if** (++a%1000==0)
13. {
14. Console.WriteLine($"{a}已接收到信息：{mes}");
15. }
17. }
18. }

**3.3.3 文件客户端类（FileClient）**

**（1）主要属性及说明：**

1）可设置属性

**ReceiveDirectory**：接收文件的默认接收目录。

2）只读属性

**UploadFileBlocks**：上传文件包集合。

**DownloadFileBlocks**：下载文件包集合。

**DownloadProgressValue**：获取下载进度。

**UploadProgressValue**：获取上传进度。

**DownloadFileInfo**：获取正在下载的文件信息。

**UploadFileInfo**：获取正在上传的文件信息。

**DownloadSpeed**：下载速度。

**UploadSpeed**：上传速度。

**（2）主要事件及说明：**

|  |  |
| --- | --- |
| 事件名 | 事件触发及描述 |
| ConnectedService | 成功连接到服务器时触发 |
| DisConnectedService | 主动或被动断开服务器时触发 |
| ReceiveSystemMes | 接收到服务器发送的系统消息 |
| BeforeReceiveFile | 给服务器请求下载文件，并且得到允许下载的回应之后，触发该事件 |
| BeforeUploadFile | 当客户端即将上传文件，但还未请求服务器时触发 |
| UploadFileFinshed | 给服务器发送完文件时触发 |
| ReceiveFileFinished | 接收完客户端上传的文件时触发 |
| TransferFileError | 传输文件有错误信息，或者收到服务器关于文件传递的信息后触发，常见的有，不允许下载时触发，请求下载文件不存在时触发，下载异常中断时触发 |

**（3）主要方法及说明：**

|  |  |  |
| --- | --- | --- |
| 方法名 | 方法功能及描述 | 参数类型及说明 |
| SendSystemMessage | 给服务器发送系统消息 | Text：表示发送内容 |
| DownloadFile | 下载文件 | url.FilePath：请求下载的文件路径  url.Restart：是否重新下载文件（在服务器开启断点续传时，可以不续传，重新开始下载） |
| UploadFile | 上传文件 | url.FilePath：请求上传的文件路径  url.Restart：是否重新上传文件（在服务器开启断点续传时，可以不续传，重新开始上传） |
| PauseDownload | 暂停下载 | 无 |
| StopDownload | 停止当前下载 | 无 |
| PauseUpload | 暂停上传 | 无 |
| ResumeUpload | 恢复上传 | 无 |
| StopUpload | 停止上传 | 无 |
| Dispose | 释放资源 | 无 |

**3.3.4 RPC客户端类（RPCClient）**

**（1）主要属性及说明：**

1）只读属性

**（2）主要事件及说明：**

|  |  |
| --- | --- |
| 事件名 | 事件触发及描述 |
| ConnectedService | 成功连接到服务器时触发 |
| DisConnectedService | 主动或被动断开服务器时触发 |

**（3）主要方法及说明：**

|  |  |  |
| --- | --- | --- |
| 方法名 | 方法功能及描述 | 参数类型及说明 |
| SendSystemMessage | 给服务器发送系统消息 | Text：表示发送内容 |
| DownloadFile | 下载文件 | url.FilePath：请求下载的文件路径  url.Restart：是否重新下载文件（在服务器开启断点续传时，可以不续传，重新开始下载） |
| UploadFile | 上传文件 | url.FilePath：请求上传的文件路径  url.Restart：是否重新上传文件（在服务器开启断点续传时，可以不续传，重新开始上传） |
| PauseDownload | 暂停下载 | 无 |
| StopDownload | 停止当前下载 | 无 |
| PauseUpload | 暂停上传 | 无 |
| ResumeUpload | 恢复上传 | 无 |
| StopUpload | 停止上传 | 无 |
| Dispose | 释放资源 | 无 |

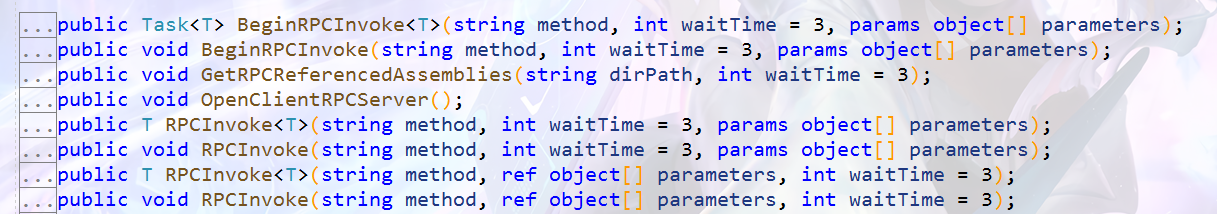
**（4）使用**

**连接服务器**

1. **static** **void** Main(**string**[] args)
2. {
4. Console.ReadKey();
6. rpcClient = **new** RPCClient();
8. ConnectSetting confirmSocket = **new** ConnectSetting();
9. confirmSocket.TargetIP = "127.0.0.1";
10. confirmSocket.TargetPort = 7789;
12. rpcClient.Connect(confirmSocket);
14. rpcClient.GetRPCReferencedAssemblies(@"C:\Users\17516\Desktop\新建文件夹\");
16. Console.WriteLine("连接成功");
17. Console.ReadKey();
19. Console.ReadKey();
20. }

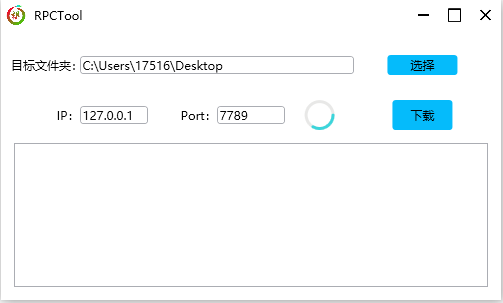
**使用方法一：**

**通过方法名+参数的方式直接调用**



**使用方法二：**

**在服务中生成代理文件，然后调用客户端的GetRPCReferencedAssemblies方法，获取代理文件，然后引用代理文件，然后使用。也可以用RPCTool工具软件下载代理文件。**



**3.3.5 文件传输类使用注意事项**

1. 最大下载速度和上传速度属性是最大的限制的值，并不是实际的传递速度，实际传递速度由网络状况决定。
2. 单线程的事件均由“接收线程”一个线程触发，所以要尽量避免在事件中处理复杂的运算，更不要让该线程形成阻塞或死循环，不然程序集将无法处理接收数据。

**四、使用其他功能**

**4.1设置最大上传和下载速度**

最大上传和下载速度均由服务器辅助类（SocketClient）的子类进行设置。

**4.2加密和解密**

程序集本身不提供加密及解密服务，所以文件，数据对象，文本等在传输过程中都是直接传输的，但是程序集里面整合了3DES的加密方法，具体使用可以直接调用DataLock的静态方法进行数据的加密和解密。

**4.3序列化和反序列化**

程序集在传递数据对象时使用的是二进制序列化，这要求被传递的类必须有[Serializable]的标签，但是程序集也提供了xml的序列化方式，可以直接调用SerializeConvert的静态方法序列化成字节数组或者字符串进行传递。

**五、致谢**

**谢谢大家对我的支持，如果还有其他问题，请加我QQ505554090。谢谢！**