JRpcPlat

版本：0.4.0
日期：2020-06-18

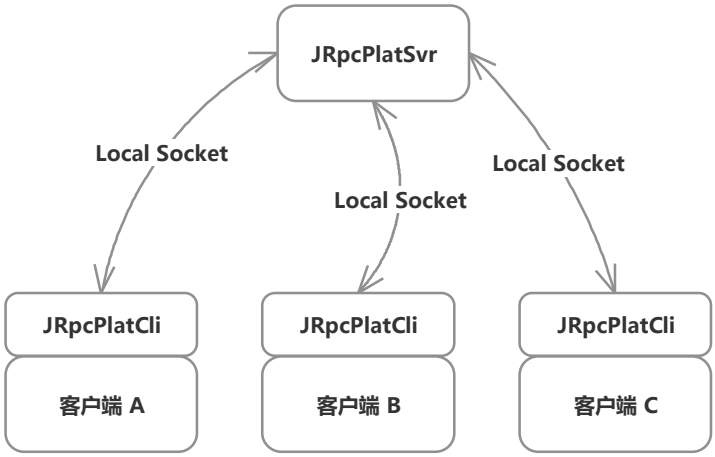
概述

JRpcPlat 是一个消息转发中间服务平台，基于 **Qt5**。它接收多个客户端连接，并保证以下述几种方式进行消息交换。
“通知消息”，“请求响应”，“二进制广播数据”。

SDK 套件分为 JRpcPlatSvr 服务端可执行程序，以及 JRpcPlatcli 动态库。

每个客户端即是使用者，也是服务者。 客户端向服务端注册服务方法字串即可为其他客户端提供方法服务。当希望使用其他客户端提供的服务时，只需要向该方法请求即可。

客户端可以动态追加或移除服务方法。



服务端

JRpcPlatSvr 启动时可以附加一些参数来强制取代配置文件中的内容。默认情况下 JRpcPlatSvr 将读取与自己处于同一路径下的名为“jrpcplat_svrdef.cfg”的配置文件。

*** 当前服务端仅支持“本地服务器”**

a) 启动时读取不同位置上的配置文件

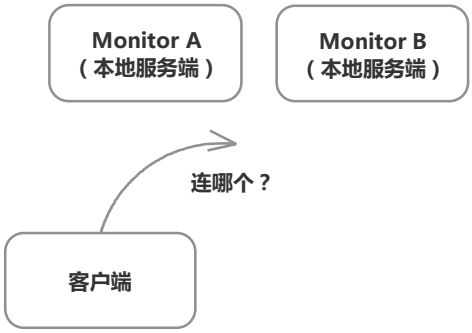
jrpcplatsvr --cfgPath="your config. file path"

配置文件路径指定后，JRpcPlatSvr 将载入指定位置上的文件，以获取描述内容。配置文件描述见其后。

b) 本地服务器的名字

为了在系统中可以使用多个本地服务端，故本地服务端是可以命不同的名字的。

jrpcplatsvr --locSvrName="server name"

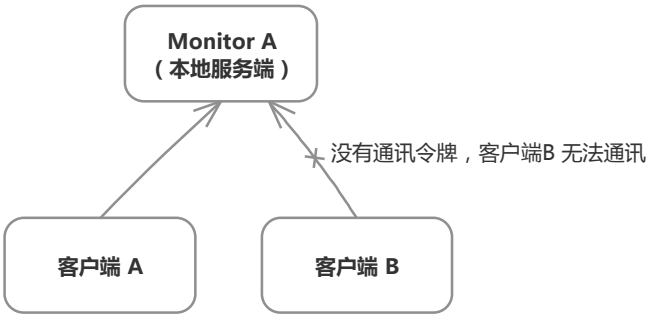


c) 本地服务器的通讯令牌

为了避免非授权应用程序进行通讯，客户端进行登记时必须带有用户自定义的通讯令牌。如果没有指定，则任何连上本地服务器的客户端都可以进行通讯。

通讯令牌是一串用户预置的字串，可以为 GUID 字串，也可以是一串乱序字符。在客户端初始化时，必须提供与服务端相同的通讯令牌字串（如果没有指定该字串，则客户端不必提供该成员）。

jrpcplatsvr --locSvrAccToken="{GUID String}"



c) 本地服务器的控制令牌

当客户端希望控制服务端的一些特别行为，则应该设置服务器一个控制令牌，只允许知道该令牌的客户端进行控制。

jrpcplatsvr --ctlToken="{GUID String}"

c-1) 退出服务端

请求服务端自动断开所有客户端并退出。请求包内容见“通讯数据”。

服务端参数配置文件

参数配置文件使用JSON格式进行描述。文件需要保存为utf-8 格式。

```
{
	"version" : "0.1",
	"timeoutOfWfAck" : 3000,
	"timeoutOfWfResp" : 30000,
	"ctlToken" : "{cdefggg-2345-0000}",

	"locSvrCfg" : {  }
}
```

version	描述当前参数配置文件的版本，当未来有所变化时，能够兼容旧有成员。
timeoutOfWfACK	描述服务端等候ACK 响应包的超时时间，默认为3000 毫秒。
timeoutOfWfResp	描述服务端等候结果包的超时时间，默认为30000 毫秒。
ctlToken	描述服务端的控制令牌字串，客户端需要提供一致的字串才能进行控制方法的操作。

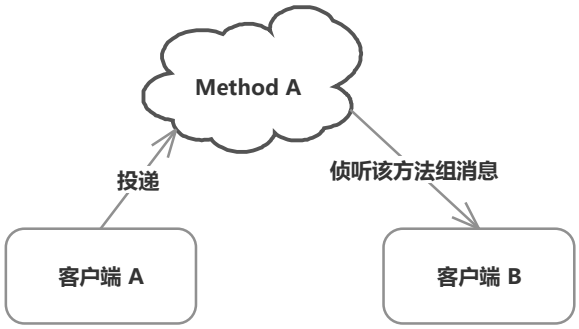
locSvrCfg 本地服务器其他配置

```
{
	"version" : "0.1",
	"accToken" : "{abcdefg-0000}",
	"svrName" : "mySvr"
}
```

version	描述当前本地配置文件的版本，当未来有所变化时，能够兼容旧有成员。
accToken	描述本地服务器的通讯令牌字串，本地客户端需要提供一致的字串才能进行通讯的操作。
svrName	本地服务器的名字，在操作系统中是唯一的

通讯交互

当客户端连上服务端后，就可以与其他客户端进行通讯。通讯都是基于方法组，任何消息都需要指定方法组名称。除了"二进制广播数据"，其他消息都使用 Json 进行描述参数。



方法

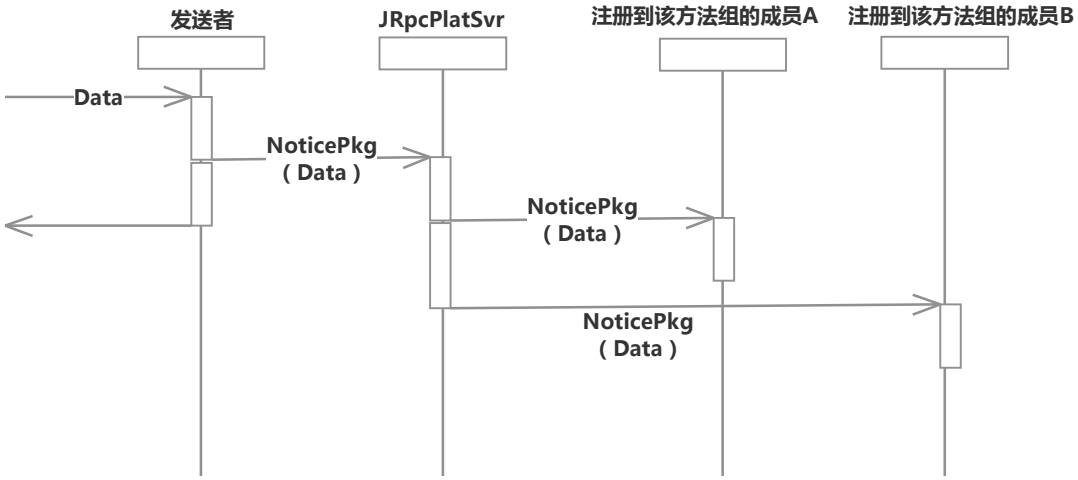
方法名在 JRpcPlatSvr 中是唯一的，可以为精确字串，也可以为模糊前缀。如："**myMethod**" 就是一个精确的方法名称，而 "**myMeth***" 就是一个以 "**myMeth**" 开头的模糊字串，* 号代表后缀为任意长度的字符串。

客户端在创建时可以注册精确方法名称，也可以注册模糊方法名称（为了侦听所有以该前缀开头的方法）。但 JRpcPlatSvr 在不同消息时处理方式是不一样的。根据下述规则进行处理：

1）（0.4.0 版本以后）对于通知消息，将发送至所有符合的精确方法组中，也会发送至所有匹配的模糊前缀的方法组中。
2）（0.4.0 版本以后）对于二进制广播数据，将发送至所有符合的精确方法组中，也会发送至所有匹配的模糊前缀的方法组中。
3）对于请求响应，将优先查看是否有精确匹配方法组，如果有，则使用精确匹配方法组中的客户端链；否则使用找到的模糊匹配的方法组中的客户端链。当前版本下，即使同时存在精确匹配的方法组 以及 模糊匹配的方法组，也只会使用精确匹配的方法组。

通知消息

发送者将通知消息包送至 JRpcPlatSvr 中，由 JRpcPlatSvr 复制到各个侦听者，发送者不用等待。



图示：通知消息的时序

二进制广播数据（0.4.0 版本以后）

发送者将二进制数据包发送至 JRpcPlatSvr 中，其他过程如同 通知消息的时序描述一样。仅是数据包装变更为 **BinBcstPkg** 。二进制广播数据与通知消息不同之处在于，通知数据参数为 Json，发送者需将参数打包而接收者需要解析参数，通知消息并不适合传送二进制数据（需要转为 Base64），而二进制广播数据带的数据为原始数据块。

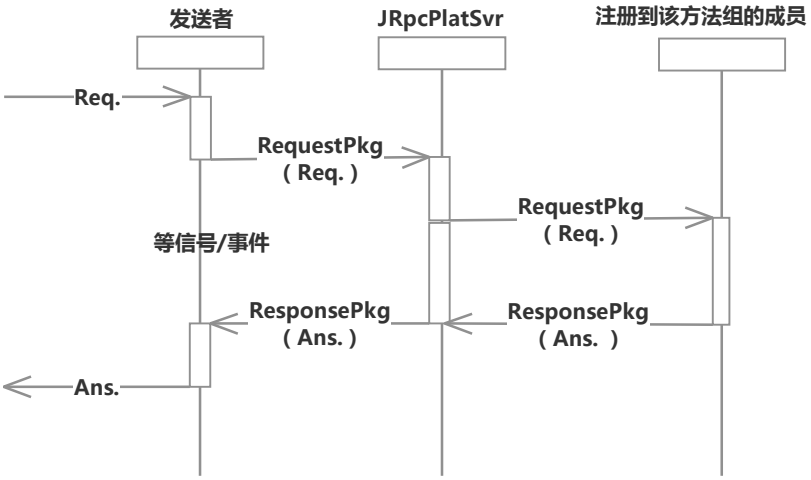
请求响应

请求响应消息模式，发送者需要等待一个结果。发送者将请求信息封装后发至 JRpcPlatSvr，由 JRpcPlatSvr 根据方法名按规则寻找到一个可以对此请求进行处理的客户端，将请求包转发，并等待其客户端处理完毕，将结果转回请求者。

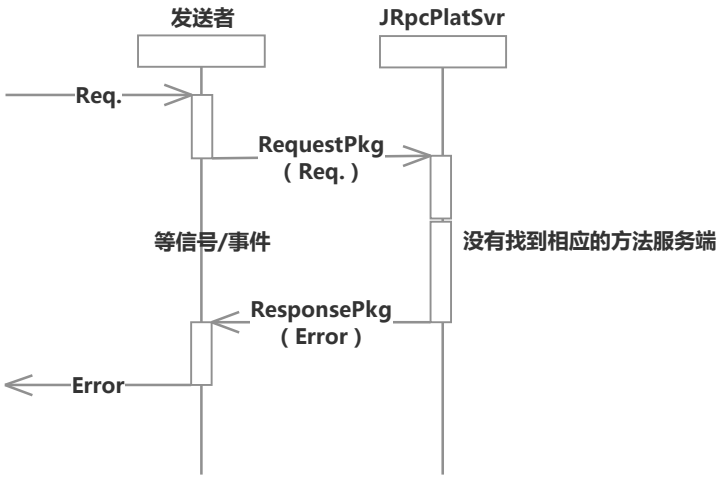
以上过程可能会产生下述几种情况：

1) 一切正常，请求者得到结果包
2) 请求者自身超时
3) JRpcPlatSvr 等候处理该请求的客户端掉线
4) JRpcPlatSvr 等候处理该请求的客户端超时（处理请求的客户端太忙，或者其他原因迟迟没有返回结果）

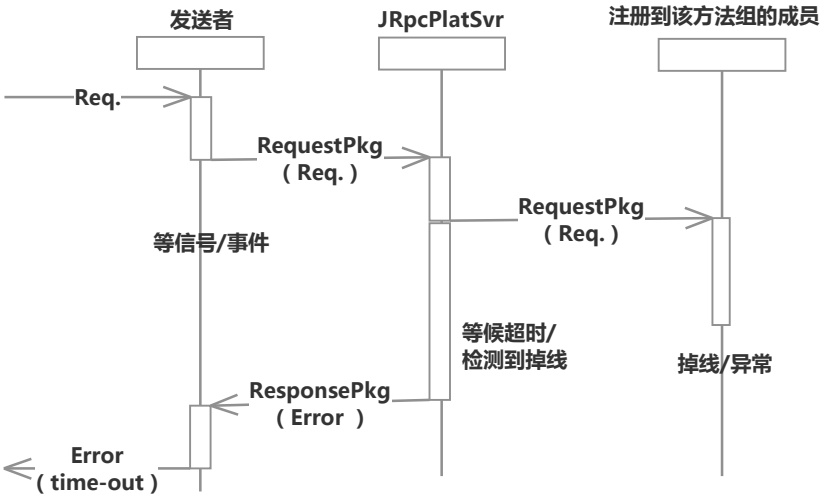
设计上，客户端在请求时会有一个超时时间，该时间可以设置为 -1 （代表无限等，直到 3，4 情形），或者 > 0 的数值（代表请求者自身计时，并且 JRpcPlatSvr 按照 JRpcPlatSvr 的配置进行超时等候 ）



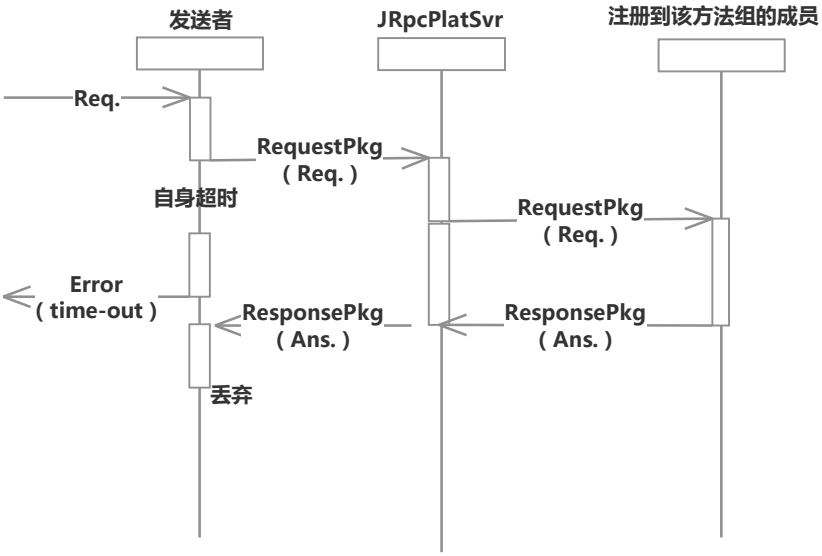
图示：请求响应正常的时序



图示：请求响应异常--不存在相应的服务端



图示：请求响应异常--服务端掉线/异常导致的超时



图示：请求响应异常--调用端直接超时

通讯数据

以下展示具体的数据包内容。

客户端进行注册时的通知消息

客户端连接上 JRpcPlatSvr 后，如果有服务能力，可以向 JRpcPlatSvr 进行注册方法。当注册完成时，JRpcPlatSvr 会固定向 "**rpc::JRpcNtc::methodReg**" 方法组广播当前客户端的注册事项。

```
NoticePkg.method  : = "rpc::JRpcNtc::methodReg"
NoticePkg.params  : = {   // 为一个  QJsonObject 对象
      "client" : {
         "type"      : "a string that the client type.";                    // 客户端的类型
         "aliasName"    :    "client's human readable name.";       // 客户端别名
      },
      "methodList" :    [    // 注册的方法列表
         "method 1", "method 2"
      ]
}
```

客户端进行注销时的通知消息

客户端可以注销 JRpcPlatSvr 上的方法JRpcPlatSvr 会固定向 "**rpc::JRpcNtc::methodUnreg**" 方法组广播当前客户端的事项。如果客户端断开链接，该消息也会通知。

```
NoticePkg.method  : = "rpc::JRpcNtc::methodUnreg"
NoticePkg.params  : = {   // 为一个  QJsonObject 对象
      "client" : {
         "type"      : "a string that the client type.";                    // 客户端的类型
         "aliasName"    :    "client's human readable name.";       // 客户端别名
      },
      "methodList" :    [    // 方法列表
         "method 1", "method 2"
      ]
}
```

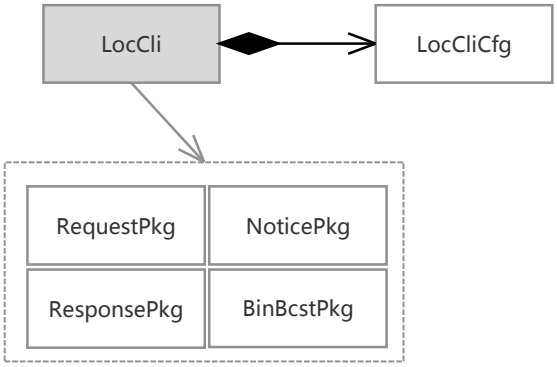
JRpcPlatSvr 的退出控制请求

由于是退出，可能该请求不会正确地返回结果，故使用该请求的客户端没有必要检查 ResponsePkg 的内容。发送该请求后，JRpcPlatSvr 将主动断开各个客户端，并结束程序执行。

```
RequestPkg.method := "rpc::JRpcCtl::exit"
RequestPkg.params :=  {
      "ctlToken" : ""     <-- 启动 JRpcPlatSvr 时定义的控制令牌字串，若没有定义，可以不存在此成员
 }
```

本地客户端 API 参考

本地客户端 **LocCli** 作为与 **JRpcPlatSvr** 进行沟通的类，同时具备发送消息/请求，接收消息/请求的能力。



图示：LocCli 的类关系图

LocCli(const LocCliCfg &)

描述：构造一个客户端对象
参数：LocCliCfg -- 本地客户端的配置类（如果没有配置，可以构造空的对象传入）

Q_SLOT void connectToServer(const QString &svr_name)

描述：连接到指定的本地服务器
参数：svr_name [IN] 本地服务器的名字。该名字在 JRpcPlatSvr 启动时提供。
注意：在连接成功后，会发出信号 **serverConnected()**。而在一切准备完毕后（注意方法等），则会引出 **ready()** 信号。
当出现无法链接时，会发出信号 **serverDisconnected()**，注意区分处理。

Q_SLOT void disconnectFromServer()

描述：断开已连接的本地服务器（可以重复调用）
注意：在断开后，会发出信号 **serverDisconnected()**。

Q_SLOT bool isWorking() const

描述：检查当前客户端是否处于正常工作状态。
返回：true 代表客户端正常工作（连上了服务端）

Q_SLOT QString serverFullName() const

描述：返回操作系统平台下的本地服务器全名

Q_SLOT bool regMethod (const QStringList &lst)

描述：注册方法到服务器，以此来为其他客户端提供服务
参数：lst [IN] -- 方法列表，所有的方法名称都列在里面
相关：
a) 在注册成功后，JRpcPlatSvr 会将客户端以及当前的方法列表广播给其他注册到 "**rpc::JRpcNtc::methodReg**" 方法的客户端。
b) **方法**代表一个组，位于此方法组的第一个客户端会收到请求消息。位于此方法组的所有客户端可以收到广播消息/通知消息

Q_SLOT bool unregMethod(const QStringList &lst)

描述：取消注册在服务器的方法。
参数：lst [IN] -- 方法列表，期望取消的方法名称都列在里面
相关：
a) 在取消注册成功后，JRpcPlatSvr 会将客户端以及取消的方法列表广播给其他注册到 "**rpc::JRpcNtc::methodUnreg**" 方法的客户端。

Q_SLOT QString uidStr( ) const

描述：返回 JRpcPlatSvr 为该客户端分配的唯一字串。

Q_SLOT bool postNotice(const NoticePkg &ntc)

描述：投递通知消息包至 JRpcPlatSvr，由 JRpcPlatSvr 进行分发
参数：ntc [IN] -- 通知消息包
返回：成功投递，返回 true。如果失败，则可能 JRpcPlatSvr 内存不足或者其他问题导致无法投递，用户应该考虑在失败时延时进行再试

Q_SLOT ResponsePkg invokeRequest(const RequestPkg &req, bool req_evt, int wait_ms = -1);

描述：向指定方法的服务端进行请求，返回结果包
参数：req [IN] -- 请求内容包
req_evt [IN] -- 是否在等候时进入事件循环（排除输入）
wait_ms [IN] -- 等候超时时间。 -1 代表无限等候；>= 0 代表客户端等候时间
返回：如果成功，返回的 ResponsePkg.isResult() 将返回 true。此时可以通过 ResponsePkg.data() 取得返回的内容。否则可以通过 ResponsePkg.errCode() 或者 ResponsePkg.errMsg() 取得错误

Q_SLOT bool postBinBcst(const BinBcstPkg &bbc)

版本：0.4.0 以后
描述：投递二进制广播包
参数：bbc [IN] -- 广播内容包
返回：成功投递，返回 true。如果失败，则可能 JRpcPlatSvr 内存不足或者其他问题导致无法投递，用户应该考虑在失败时延时进行再试

Q_SLOT bool postResponse(const ResponsePkg &resp)

描述：如果客户端作为服务端取得了请求包，这里用于回应请求包
参数：resp [IN] -- 根据请求包生成的响应包
返回：成功将响应包投递，返回 true。如果失败，则可能 JRpcPlatSvr 内存不足或者其他问题导致无法投递，用户应该考虑在失败时延时进行再试

Q_SLOT RequestPkg takeNextPendingRequest()

描述：按顺序取出等待处理的请求包
返回：请求包。
注意：需要检查返回的 RequestPkg.isEmpty() 是否为true。如果为true 表示没有等待处理的请求包
相关：在收到 **newRequest()** 信号后调用此函数有效。
警告：请求包需要在短时间内处理，并且返回 ResponsePkg。否则将导致请求者超时！

Q_SLOT NoticePkg takeNextPendingNotice()

描述：按顺序取出等待处理的通知包
返回：通知包。
注意：需要检查返回的 NoticePkg.isEmpty() 是否为true。如果为true 表示没有等待处理的通知包
相关：在收到 **newNotice()** 信号后调用此函数有效

Q_SLOT BinBcstPkg takeNextPendingBinBcst()

描述：按顺序取出等待处理的二进制广播包
返回：广播包。
注意：需要检查返回的 BinBcstPkg.isEmpty() 是否为true。如果为true 表示没有等待处理的广播包
相关：在收到 **newBinBcst()** 信号后调用此函数有效

Q_SIGNAL void newRequest ( )

描述： 当有新的请求包到达时，此信号被激发

Q_SIGNAL void newNotice ( )

描述： 当有新的通知包到达时，此信号被激发

Q_SIGNAL void serverConnected( )

描述： 服务器已连接上时，此信号被激发

Q_SIGNAL void serverDisconnected( )

描述： 服务器已断开连接时或连接到服务器失败时，此信号被激发

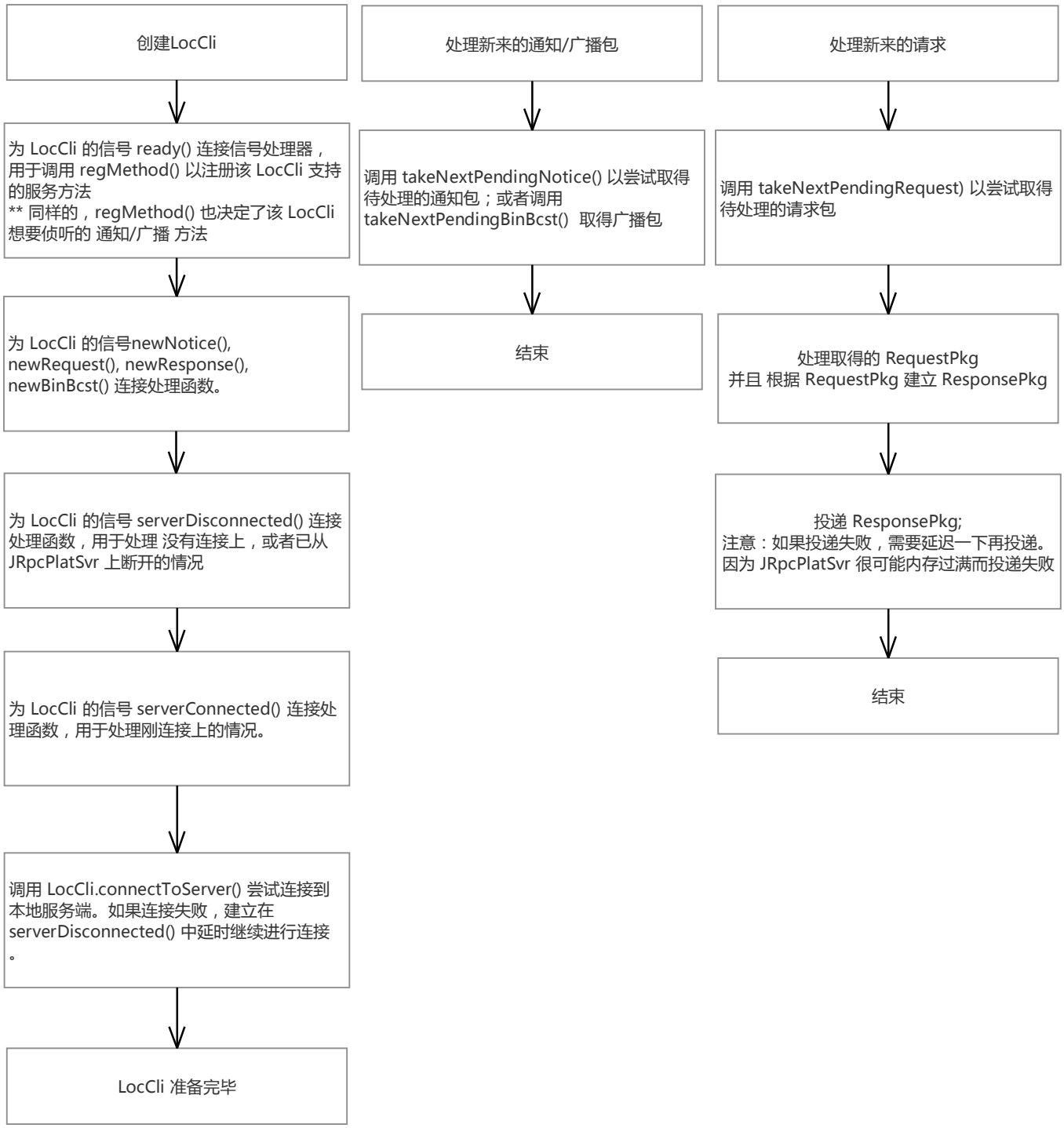
Q_SIGNAL void ready()

描述： 当一切准备好，即连接上，并且接收完服务器一些内部配置时，此信号被激发
注意：此信号发生在 serverConnected() 之后

Q_SIGNAL void newBinBcst( )

描述： 当有新的二进制广播包到达时，此信号被激发

以下使用的基本流程。


创建LocCli
为 LocCli 的信号 ready() 连接信号处理器，用于调用 regMethod() 以注册该 LocCli 支持的服务方法
** 同样的，regMethod() 也决定了该 LocCli 想要侦听的 通知/广播 方法
为 LocCli 的信号newNotice(), newRequest(), newResponse(), newBinBcst() 连接处理函数。
为 LocCli 的信号 serverDisconnected() 连接处理函数，用于处理 没有连接上，或者已从 JRpcPlatSvr 上断开的情况
为 LocCli 的信号 serverConnected() 连接处理函数，用于处理刚连接上的情况。
调用 LocCli.connectToServer() 尝试连接到本地服务端。如果连接失败，建立在 serverDisconnected() 中延时继续进行连接。
LocCli 准备完毕
处理新来的通知/广播包
调用 takeNextPendingNotice() 以尝试取得待处理的通知包；或者调用 takeNextPendingBinBcst() 取得广播包
结束
处理新来的请求
调用 takeNextPendingRequest) 以尝试取得待处理的请求包
处理取得的 RequestPkg
并且 根据 RequestPkg 建立 ResponsePkg
投递 ResponsePkg;
注意：如果投递失败，需要延迟一下再投递。因为 JRpcPlatSvr 很可能内存过满而投递失败
结束