**内部文件**

**秘密**

**文件与数据库部分系统入门指南和规范**

**平台部分**

**成都世纪顶点科技有限公司**

**2018.04.19**

**目 录**

[第一章 环境配置 5](#_Toc511921563)

[1.1安装QT 5](#_Toc511921564)

[1.2安装ICE 5](#_Toc511921565)

[1.3下载平台代码 5](#_Toc511921566)

[第二章 客户端代码解读 6](#_Toc511921567)

[2.1DrcClient流程分析 6](#_Toc511921568)

[2.1.1ChinArkDRCCLient.cpp 客户端程序的入口 7](#_Toc511921569)

[2.1.2DRCUtil\* util = DRCUtilFactory::GetDRCUtil(); 7](#_Toc511921570)

[2.1.3 clientInit() ; 7](#_Toc511921571)

[2.1.4util->initPlatform(); 8](#_Toc511921572)

[2.1.5 util->drc\_access(::g\_iceConfigPath.c\_str(), F\_OK) 8](#_Toc511921573)

[2.1.6 ::g\_lisenceHandller = new LisenceHandller; 8](#_Toc511921574)

[2.1.7 startClientListner(ic, argc, argv); 8](#_Toc511921575)

[2.1.8 while(!::g\_isClientExit) 8](#_Toc511921576)

[2.1.9跳入主线程 8](#_Toc511921577)

[2.2ICE服务使用 8](#_Toc511921578)

[2.2.1服务绑定 9](#_Toc511921579)

[2.2.2服务调用 9](#_Toc511921580)

[2.3config.exe 程序部分 10](#_Toc511921581)

[2.3.1run() 10](#_Toc511921582)

[2.3.2. restartService() 11](#_Toc511921583)

[2.4主线程 11](#_Toc511921584)

[2.4.1m\_utilInstance = DRCUtilFactory::GetDRCUtil(); 13](#_Toc511921585)

[2.4.2开启相关线程 13](#_Toc511921586)

[2.4.3获取服务端对象 14](#_Toc511921587)

[2.4.4注册client回调服务 14](#_Toc511921588)

[2.4.5向服务端注册 14](#_Toc511921589)

[2.4.6循环检测 15](#_Toc511921590)

[2.4.7等待并关闭所有线程 16](#_Toc511921591)

[2.5授权检测线程 17](#_Toc511921592)

[2.5.1原有的部分 17](#_Toc511921593)

[2.5.2现有的部分-通过watchdog 17](#_Toc511921594)

[2.6控制消息线程 18](#_Toc511921595)

[2.7配置消息线程 19](#_Toc511921596)

[2.8保活线程 20](#_Toc511921597)

[2.9客户端回调类ClientCallbackI 21](#_Toc511921598)

[第三章 服务端代码解读 22](#_Toc511921599)

[3.1DRCServer.cpp主进程 22](#_Toc511921600)

[3.2service.cpp主线程 23](#_Toc511921601)

[3.3客户端离线清除线程 23](#_Toc511921602)

[3.4归档线程 24](#_Toc511921603)

[3.5日志更新线程 25](#_Toc511921604)

[3.5.1SystemLogUpdater线程 25](#_Toc511921605)

[3.5.2JobLogUpdater线程 25](#_Toc511921606)

[3.6服务回调类TaskControlI 25](#_Toc511921607)

[第四章 编译源码 32](#_Toc511921608)

[4.1DrcClient和DrcServer的编译步骤 32](#_Toc511921609)

[4.1.1源码下载 32](#_Toc511921610)

[4.1.2在本地配置工程运行的环境 32](#_Toc511921611)

[4.1.3window端 32](#_Toc511921612)

[4.1.4linux端 32](#_Toc511921613)

[4.2需要哪些组件/依赖才能编译 33](#_Toc511921614)

[4.3编译的操作系统环境 33](#_Toc511921615)

[第五章 总结 34](#_Toc511921616)

[5.1整体结构的数据流图 34](#_Toc511921617)

[5.2客户端内容一览 34](#_Toc511921618)

[5.3服务端一览 34](#_Toc511921619)

第一章 环境配置

1.1安装QT

安装qt-opensource-windows-x86-vs2010-4.8.6.exe

搭建QT环境。(对路径没有特殊要求)

1.2安装ICE

安装ICE库。

安装路径：D:\Program Files (x86)\ZeroC\Ice-3.5.1

1.3下载平台代码

通过SVN->switch->svn://192.168.0.91/v2platform/dev

(下载之前请确保有访问SVN代码库的权限)

至此在dev/client/DRCClient.sln就整个项目的入口。

第二章 客户端代码解读

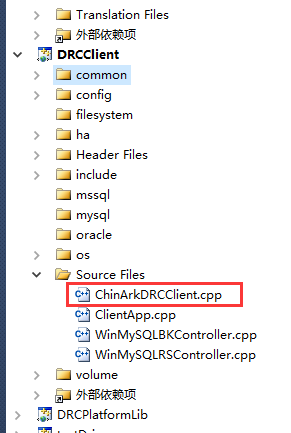
2.1DrcClient流程分析



客户端主进程大体流程

详细介绍：

2.1.1ChinArkDRCCLient.cpp 客户端程序的入口



入口文件位置

2.1.2DRCUtil\* util = DRCUtilFactory::GetDRCUtil();

定义一个util对象，此处的util对象中实现了多种需要获取系统上下文环境的方法。

2.1.3 clientInit() ;

初始化内容包括：

1.初始化临时目录

2.初始化bin路径

3.初始化“控制消息”路径

4.初始化“任务”路径

5.初始化“时间配置”文件­

2.1.4util->initPlatform();

主要进行ssl连接的一些初始化。

2.1.5 util->drc\_access(::g\_iceConfigPath.c\_str(), F\_OK)

读取ice框架的配置路径。

因为本程序是基于ICE框架的，所以程序运行前需要保证ICE库的路径正确。

2.1.6 ::g\_lisenceHandller = new LisenceHandller;

::g\_lisenceHandllerThreadControl = ::g\_lisenceHandller->start();

授权检测线程的开启，具体内容详见: 2.5授权检测线程。

2.1.7 startClientListner(ic, argc, argv);

启动客户端监听服务。

此处客户端的监听部分是标准的ICE服务绑定规范，主要作用是实现服务端回调客户端内容。客户端提供了大量的回调方法。详细的ICE绑定方式请见

2.1.8 while(!::g\_isClientExit)

客户端主线程循环检测。

主要是为了检测Config.exe这个QT部分的修改、退出、重启服务的变化。时间间隔为5s。

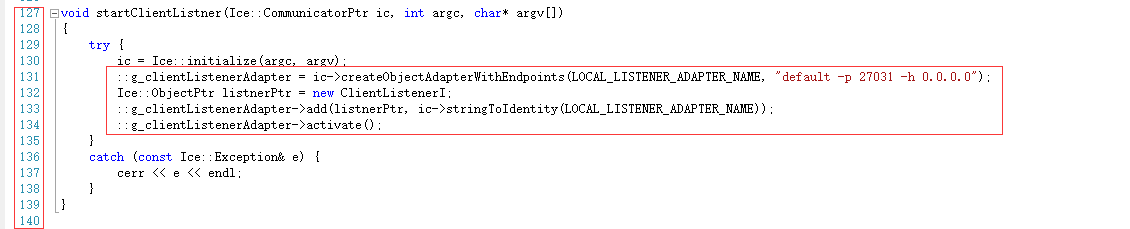
2.1.9跳入主线程

app.main(argc, argv, ::g\_iceConfigPath.c\_str()即转到ClientApp中。

2.2ICE服务使用

平台上回调服务主要是通过ICE框架来实现的，以startClientListner为例进行说明。

2.2.1服务绑定



描述：

1.ice进行初始化

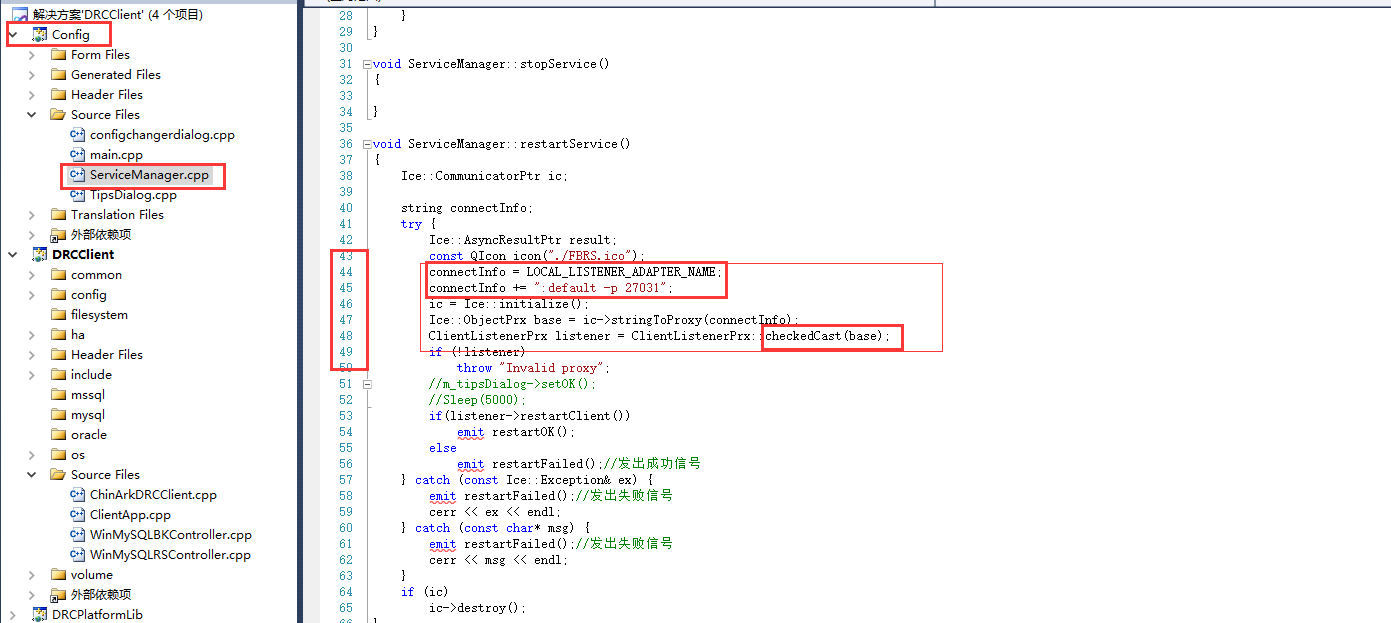
2.注册ice服务名称与设置默认参数

3.声明ice服务实际处理对象

4.适配器与服务对象进行绑定

5.激活服务

2.2.2服务调用

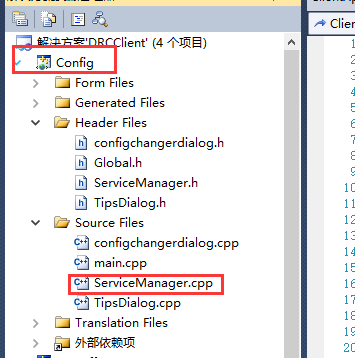


描述：

1. 构造服务名称与参数
2. Ice初始化
3. 通过stringToProxy由名称映射服务对象
4. 通过checkedCast()获取对象实例

2.3config.exe 程序部分

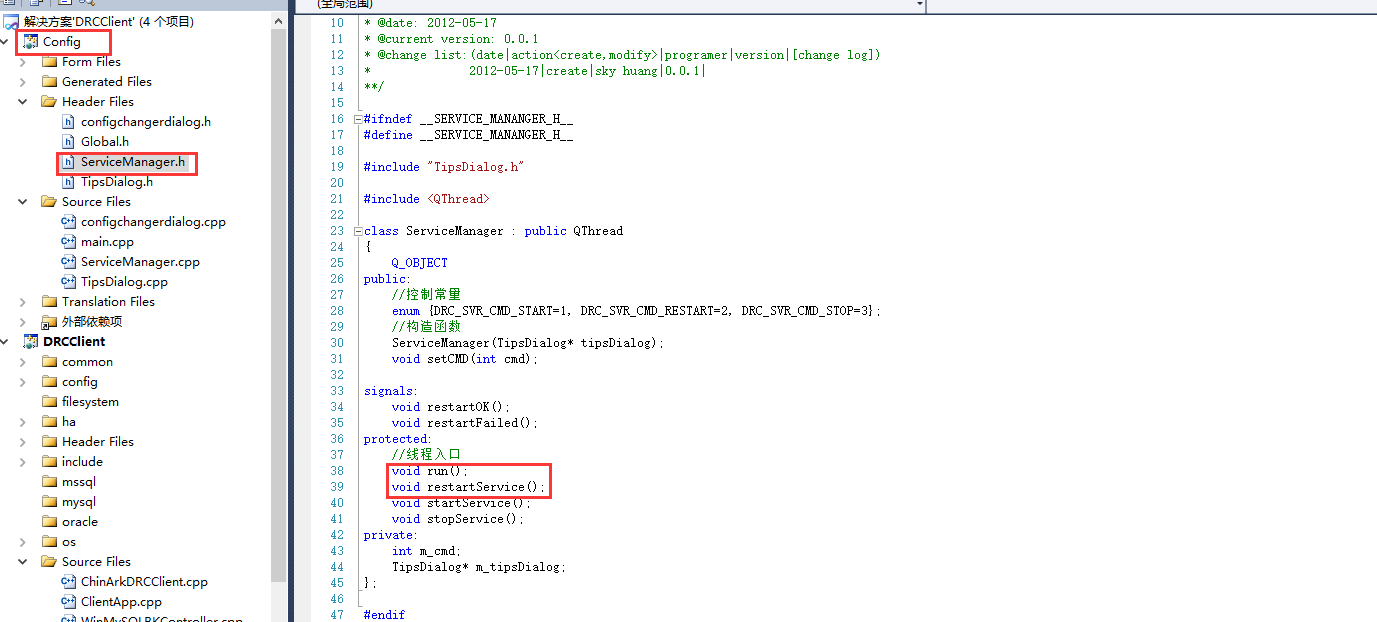
就是填写IP，开启服务的GUI客户端部分。代码位置：



描述：

除了大部分图形化界面之外，主要的业务逻辑是在ServiceManager.cpp中

这是一个线程，不断的监听config.exe命令消息的变化



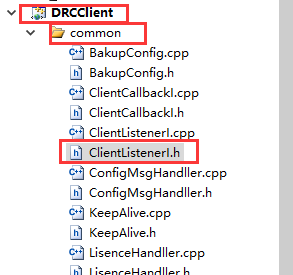
2.3.1run()

功能：线程的启动入口

判读当前需要处理的标志，对应开启停止服务、重启服务、开启服务的功能。

2.3.2. restartService()

主要做的是判断重启成功与否，重启服务的实际内容已经被ice底层实现。重启服务在ClientListener.h中

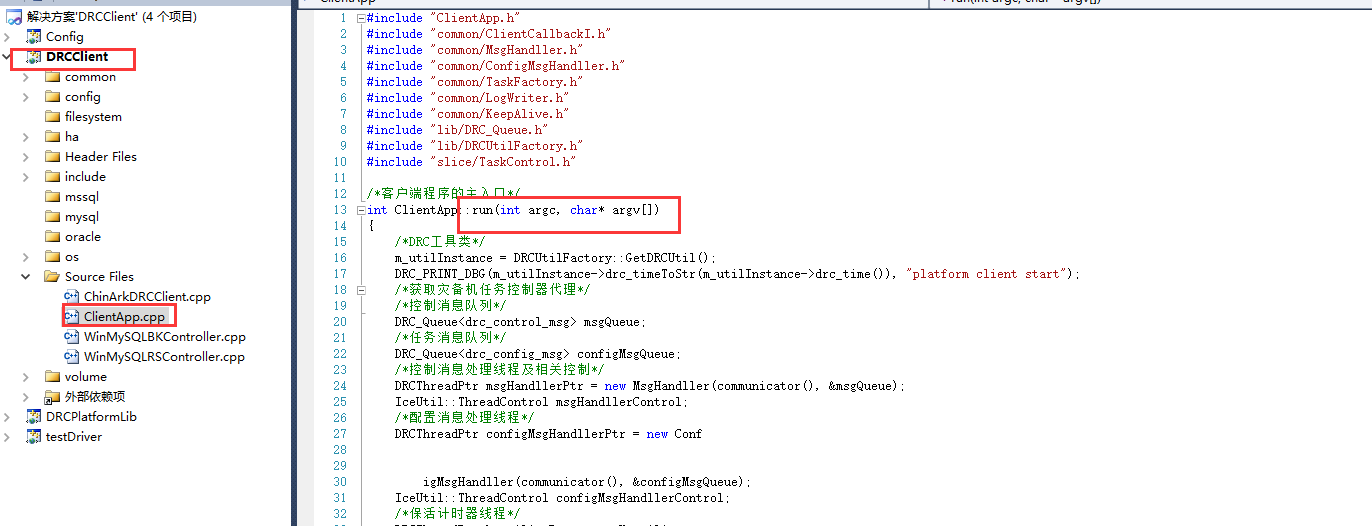


功能：重启客户端

描述：当收到“重启服务”的命令后，回调客户端监听类，实现服务重启。如果判断重启成功则返回true。

2.4主线程

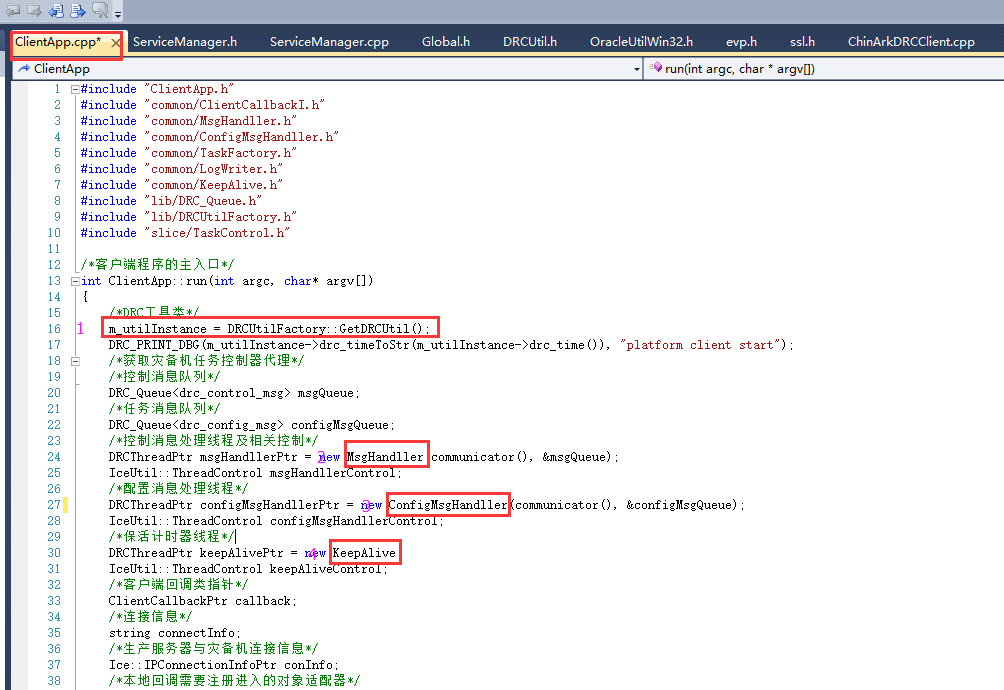
主线程位于ClientApp.cpp





主线程主要逻辑

详细介绍：

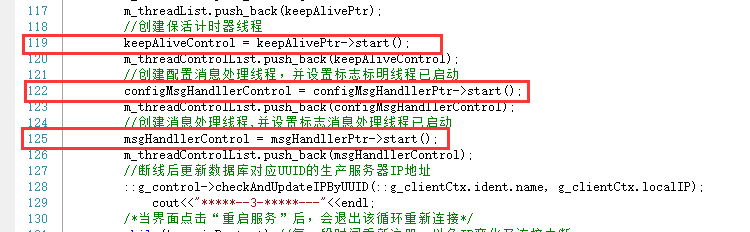


2.4.1m\_utilInstance = DRCUtilFactory::GetDRCUtil();

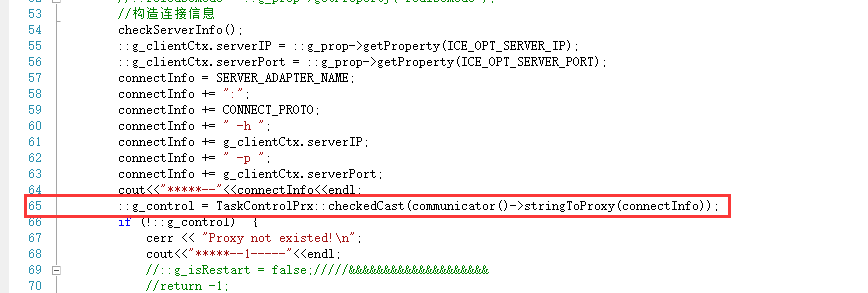
获取DRCUtil对象，这是一个工具类

2.4.2开启相关线程

分别注册了三个线程分别是MsgHandler（控制消息处理线程）、ConfigMsgHandler（配置消息处理线程）、KeepAlive（保活线程）具体的三个线程的内容详见:2.6控制消息线程、2.7配置消息线程、2.8保活线程。

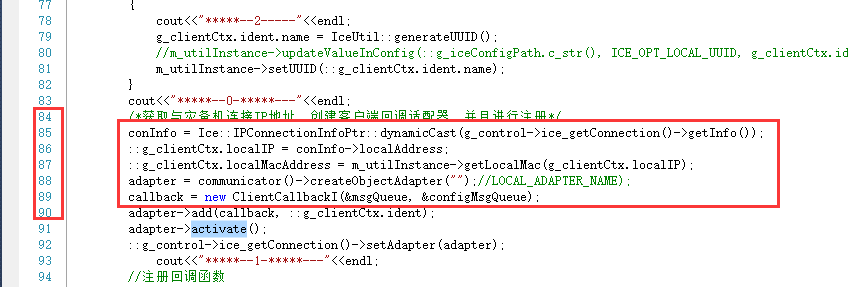


2.4.3获取服务端对象



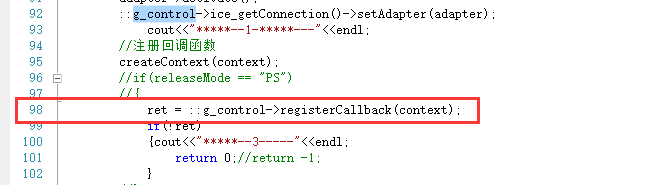
获取server端的回调服务，可以请求server部分的函数功能

2.4.4注册client回调服务



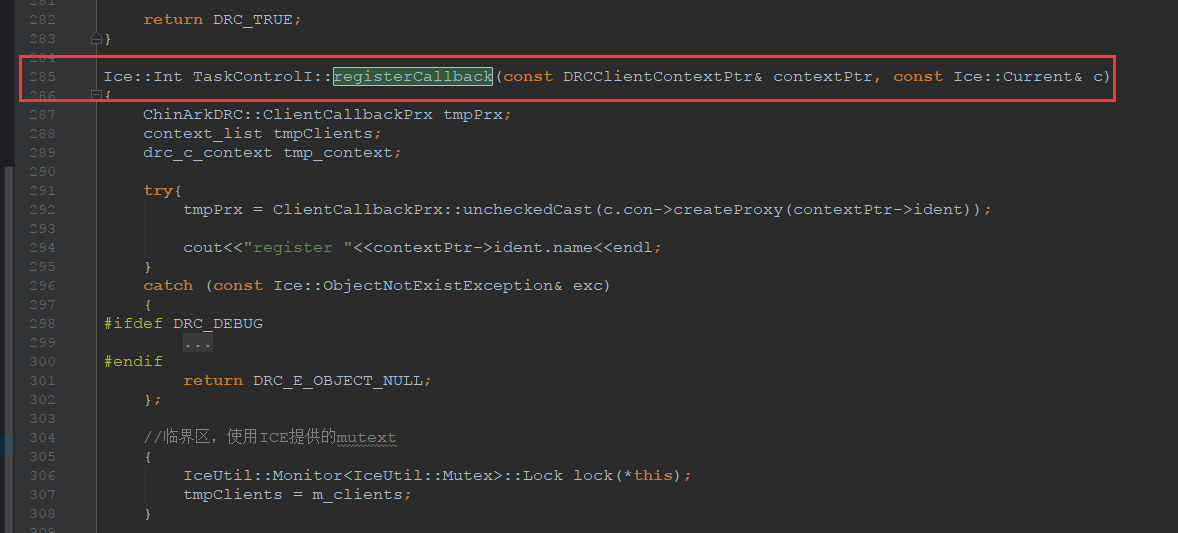
注册client端的回调服务，以便server端进行调用，服务名称为对应的UUID

2.4.5向服务端注册

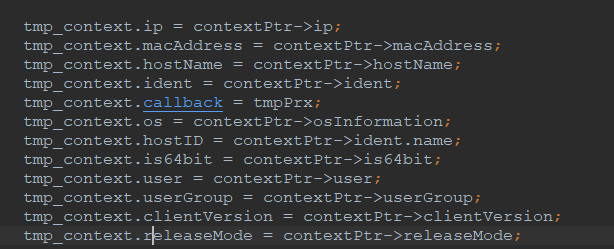


功能：向服务端进行注册。

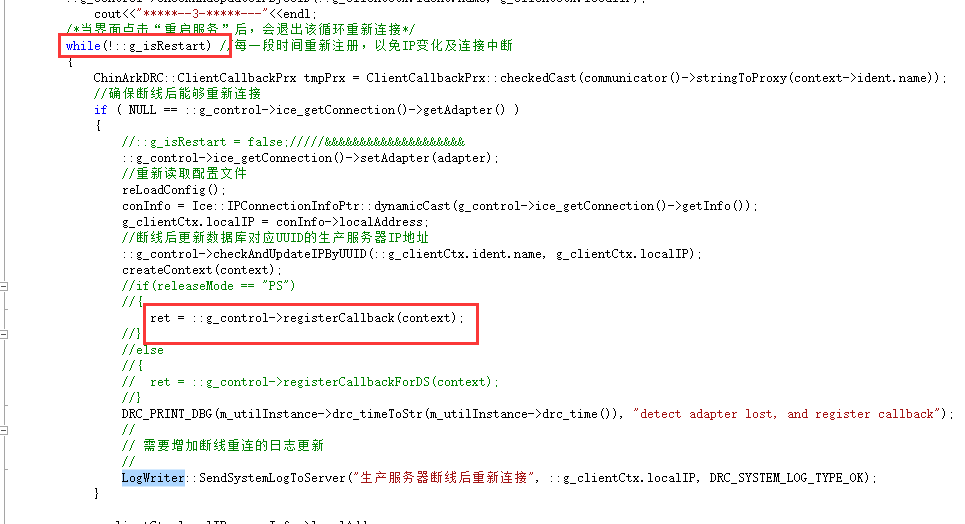
描述：



首先监测该客户端是否已注册，如果已注册，则退出；否则构造相关信息，进行注册。构造的信息如下：



2.4.6循环检测



功能：通过一定时间间隔的重新注册，来实现断线重连、IP变化的异常处理。

2.4.7等待并关闭所有线程



功能：完成主线程退出的过程。

描述：关闭的过程，将所有子线程关闭，销毁连接器，将服务端回调对象置空等操作。

详细描述：



遍历线程列表，将列表中的线程依次关闭。

2.5授权检测线程

2.5.1原有的部分



功能：检测模块是否被授权

描述：通过服务端向客户端的信息中更新授权的编码，线程通过不断检测编码的状态，以实时的更新授权的内容，开启或者关闭对应的模块。

2.5.2现有的部分-通过watchdog



通过上述函数，从授权文件中读取相关的授权信息，对授权的模块进行开启与关闭。

授权过程:

1.判断当前是否仍然有执行的任务，如果有应该先停止后才能进行授权操作。

2.在前台界面点击所要选择的模块，通过form表单传给后台的addProductHost.php文件。

3.根据传回的Module信息，利用checkLicence()方法判断是当前是否有剩余的licence以供授权。

4.如果可以授权，则利用drcSendConfigMsgSync()方法使生产服务器同步新的授权信息，通过addProductHostConfig()方法，写入到Config.php中。

5.向ProductHost表中更新或者插入这条新的记录。

2.6控制消息线程



功能：通过创建文件的方式进行消息处理与程序控制，与主线程共享控制消息队列。

描述：主要的处理函数为：handleOldVersionControlMsg；其中主要的控制消息为：启动、停止、暂停、恢复、删除等命令。

详细介绍：

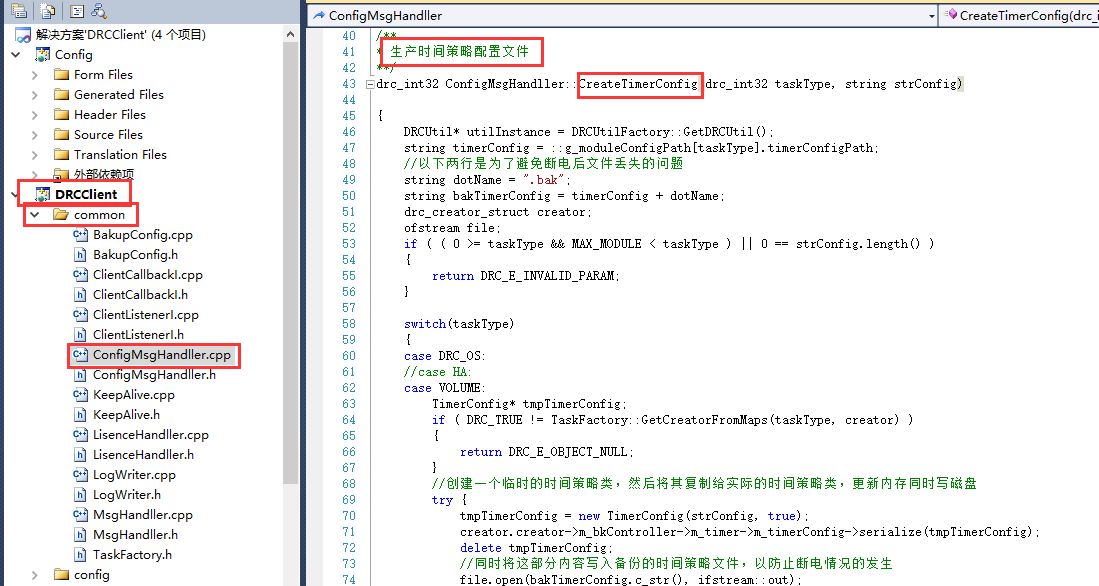
1.线程在没有收到客户端退出的消息时，一直存在。

2.利用从和主线程共享的消息队列，每次从消息队列中获取一条消息。

3.利用HandleControlMsg方法，根据这条消息的类型，命令，消息。在相应的文件路径下创建/删除相应名称的文件，通过对应的文件来实现消息传递和控制。

4.通过不断的循环，实现对消息的持续处理。

2.7配置消息线程



功能：配置消息主要是指时间策略，即什么时间开始备份，每隔多久时间重复备份，与主线程共享配置消息队列。

描述:将这些时间消息写入时间策略文件中，客户端的php脚本通过获取时间策略文件，自行进行时间的计算。

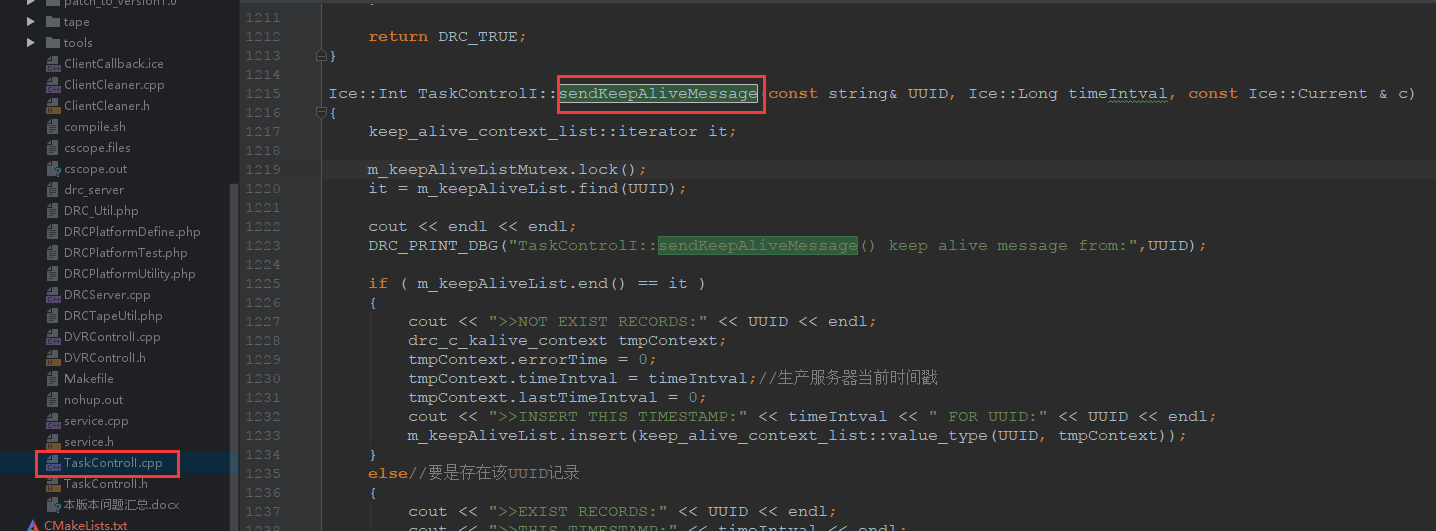
2.8保活线程



功能：通过服务端的sendKeepAliveMessage()不断的更新一个时间戳，从而告诉服务器，客户端在线

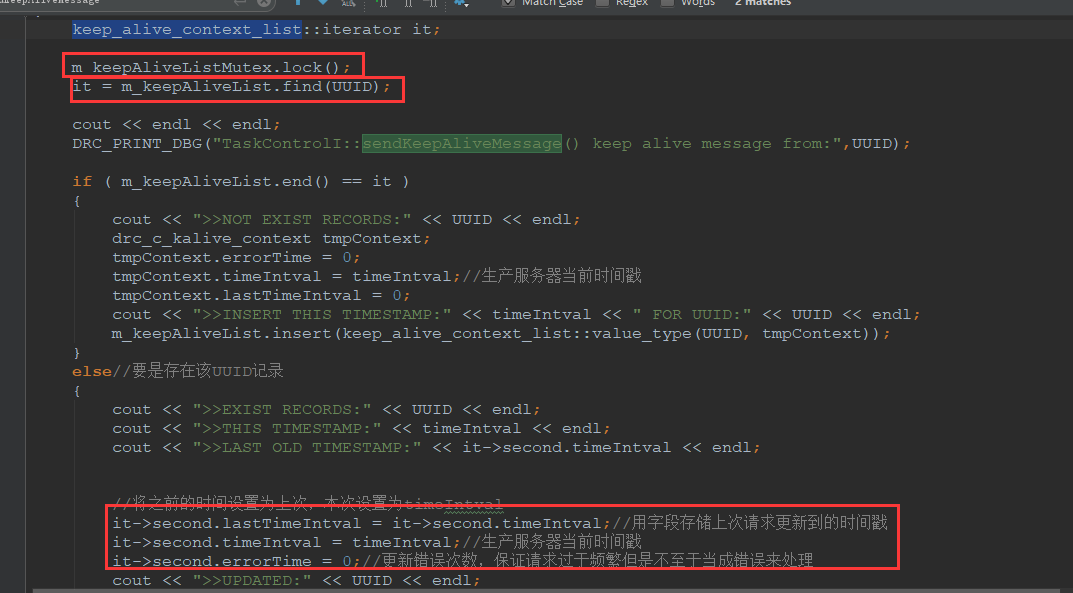
描述：服务端维护着一个在线列表，通过不断的比对时间戳，如果客户端的时间戳一定时间未更新则判定客户端下线。所以该线程与客户端的在线监测部分，共同构成在线检测模块。

此方法是通过服务端的回调函数实现的，具体位置：



详细介绍：

sendKeepAliveMessage功能介绍：



1. 对保活列表进行加锁
2. 在保活列表中寻找是否存在该客户机。
3. 如果没有则证明首次登录，如果有则更新时间戳与错误累计计数则可。

2.9客户端回调类ClientCallbackI

ClientCallbackI提供的功能有哪些：

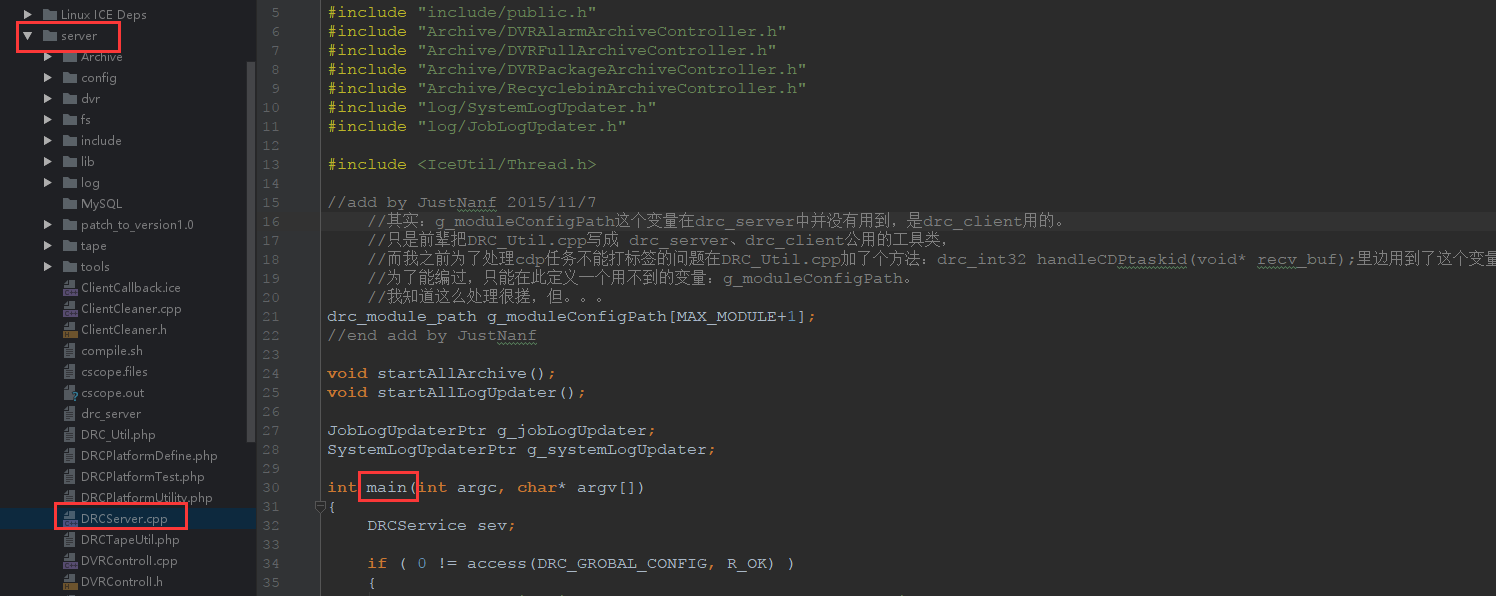
1.关于任务的处理方法：获取controlmsg/获取configmsg/获取下次任务开始时间/删除生产服务器/删除任务/检查相关模块是否激活。

2.关于本地的处理方法：获取客户端根路径/检查本地磁盘空间/获取磁盘最大空闲容量/获取真实IP/执行shell命令并返回结果。

3.关于数据库的处理方法：获取数据库实例信息。对于sqlserver额外有检查实例中数据库是否存在，检查恢复目录是否存在的方法。

第三章 服务端代码解读

3.1DRCServer.cpp主进程

程序入口：



主程序主要流程

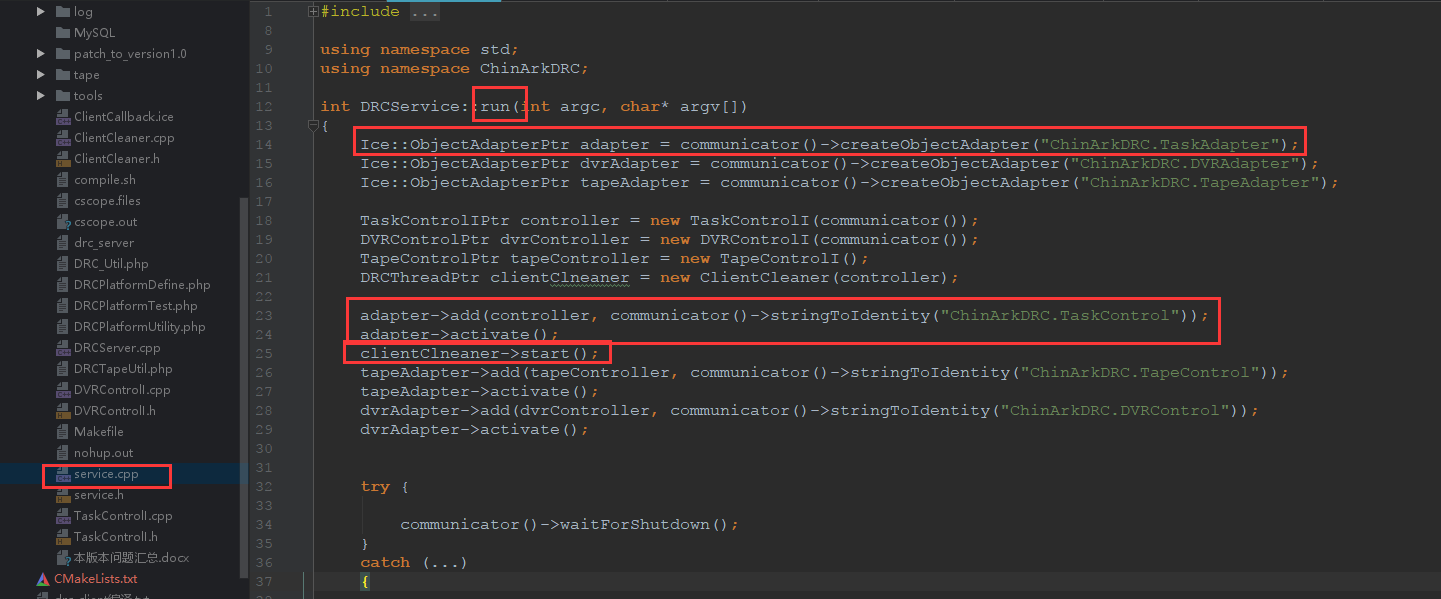
功能：服务器程序入口。

描述：1.开启各类型归档线程

2.开启日志更新线程。

2.开启service主线程。

3.2service.cpp主线程

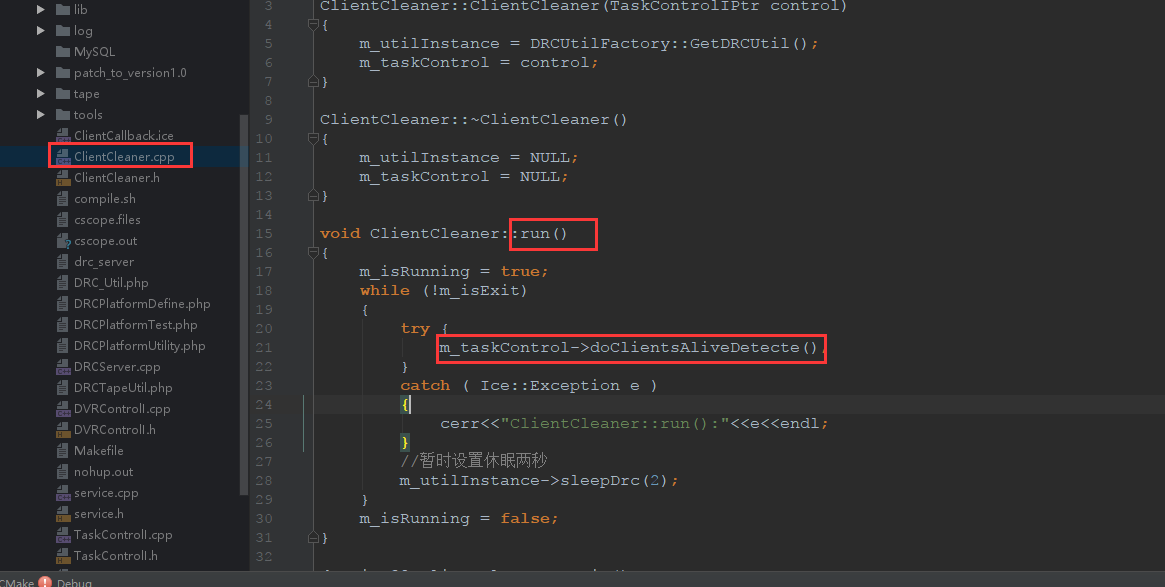


功能：注册相关的服务。

描述：1.注册TaskControlI服务。

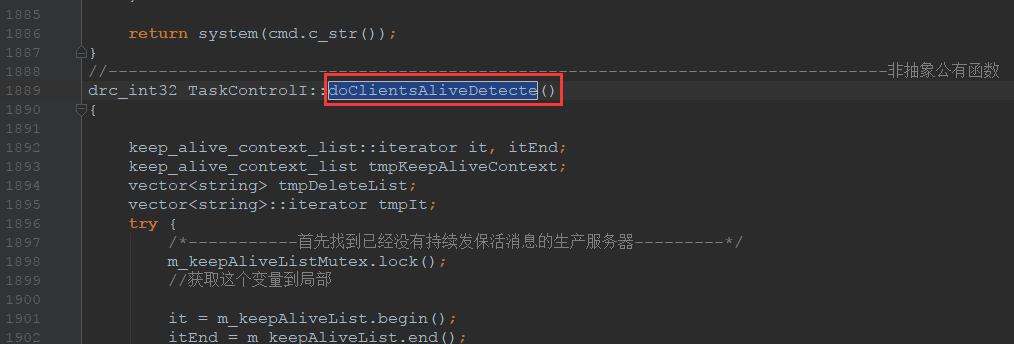
2.启动客户端离线清除线程。

3.3客户端离线清除线程



功能：检测离线客户端。

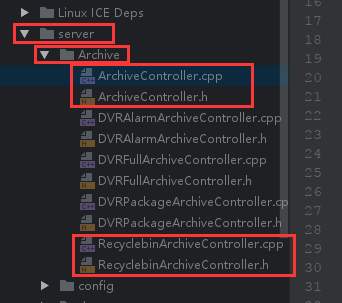
描述：将离线的客户端从在线客户端列表中清除。



详细介绍：

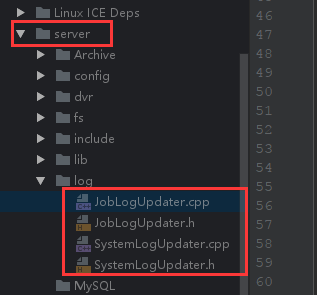
不断的比对当前时间戳与上一个时间戳是否相同。若相同则代表时间戳未更新。若连续5次未更新则判定为断线。

3.4归档线程



功能：删除多余的备份集，只保留有限个备份集，以节省空间。

3.5日志更新线程



3.5.1SystemLogUpdater线程

1. 从系统日志m\_logList获取一条记录。
2. 将这条系统日志信息插入到数据库SystemLog这个表中。

3.5.2JobLogUpdater线程

1.从工作日志m\_logList获取一条记录。

2.利用这条记录的信息对数据库进行一次查询操作，获取日志相关信息，生产服务器相关信息，任务相关信息。

3.如果查询成功，则向数据库中JobLog表中插入一条日志记录。

3.6服务回调类TaskControlI

TaskControl中的一个基本思路：

1. 从m\_clients中根据UUID查找到对应的生产服务器。
2. 利用客户端回调函数调用对应的客户端的方法，实现真正的业务部分。

详细介绍：

|  |  |  |
| --- | --- | --- |
| 模块名称 | 模块功能 | 模块描述 |
| 消息模块 | | |
| updateStartTime | 更新JobInfo中的JobStime |  |
| sendControlMsg | 实现向客户端发送控制消息的功能 | ClientCallBack.getControlMsg()将ControlMsg放入m\_msgQueue中 |
| sendControlMsgSync | 直接让客户端将ControlMsg处理 | ClientCallBack.getControlMsgSync()其中的主要方法是HandleControlMsg，其主要功能是：根据相关的命令创建对应消息名称的文件。  ControlMsg主要包括：  pause/delete/stop/execute/resume/  rs/rs\_pause/rs\_stop/rs\_resume/ 等之类信息 |
| sendConfigMsgSync | 客户端处理ConfigMsg | ClientCallBack.getControlMsgSync()其中的主要方法是bakupConfigCtrl  之后会调用bakupConfigType或者restoreConfigType这些方法会将ConfigMsg保存到生产服务器上 |
| sendConfigMsg | 发送ConfigMsg到生产服务器 | ClientCallBack.getConfigMsg()将ConfigMsg放入m\_configMsgQueue中 |
| writeContentToFile | 写文件 |  |
| 生产服务器相关模块 | | |
| registerCallback | 注册 | 在m\_clients插入一台生产服务器的context |
| unregister | 退出 | 将m\_clients中对应的UUID取出队列，实现退出 |
| checkAndUpdateIPByUUID | 更新ProductHost中UUID对应的IP |  |
| getAllOnlineClients | 获取所有在线的生产服务器 | 将m\_clients表中的数据拷贝到list中 |
| isClientOnline | 查找是否在线 | 在m\_clients查找对应的UUID |
| getClientInfo | 查找在线客户端的相关信息 | 从m\_clients中根据UUID查找相关的生产服务器信息 |
| getClientConfig | 从ProductHost表中查询相关的客户端信息 |  |
| requestDeleteBackupData | 利用shell脚本将灾备机备份的对应UUID的数据删除 |  |
| wakeUpClient | 唤醒客户端 |  |
| deleteTargetProductHost | 删除生产服务器 |  |
| deleteTargetJob | 删除taskPath/tmpPath/controlPath,从而来删除任务 |  |
| sendKeepAliveMessage | 更新lastTimeIntval和timeIntval |  |
| getOSVolume | 获取系统卷信息 |  |
| wakeUpPowerOffHost | 通过mac地址，开启对应的主机。 | 前提：网卡是带电的，可以设置。 |
| modifyBootMenu | 更改BootMenu路径 |  |
| deleteOSBootConfig | 删除BootConfig路径 |  |
| doClientsAliveDetecte | 检测生产服务器是否在线；如果不在线，则将该生产服务器从m\_clients中删除。 | 这个方法会被ClientCleaner线程循环调用，进而维护在线生产服务器列表  过程：1.比对timeIntval与lastTimeIntval是否相等。如果相等，代表时间未更新。将错误标志errorTime加一。  2.如果错误标志errorTime累计到5，则将该生产服务器从在线列表中删除。 |
| getTargetHostTime | 获取当前时间 |  |
| getTaskNextRunTime | 获取下次任务开始的时间 |  |
| getHostNameByUUID | 从ProductHost表中根据UUID查询出生产服务器的名称。 |  |
| getDrcClientTrueIpByUUID | 利用UUID回调ClientCallBack，读取本地的配置文件，获取真正的IP。 |  |
| 日志模块 | | |
| sendJobLog | 插入一条任务日志 |  |
| sendSystemLog | 插入一条系统日志 |  |
| changeLogFilter | 更改日志过滤器 |  |
| loadLogFilter | 加载日志过滤文件 | 给文件中可以设置任务日志过滤和系统日志过滤。目的就是保留需要级别的日志。 |
| testJobLogCode | 过滤任务日志 |  |
| writeSystemLog | 写系统日志 |  |
| 工具模块 | | |
| doShellCMDWithResult | 执行shell并将结果保存到对应的文件中 |  |
| doShellCMDWithoutResult | 执行shell |  |
| sendURLtoDrcClientAndGetExeResult | 给客户端发送URL |  |
| sendExecShellCommand | 向客户端发送Shell命令，并由客户端执行 |  |
| 文件备份模块 | | |
| getClientFileSystem | 获取客户端文件系统 |  |
| checkTaregtDiskSpace | 获取磁盘剩余容量 |  |
| 数据库模块 | | |
| getSQLInfoFormClient | 获取SQL Server信息 | 回调ClientCallBackI中getAllSQLServerInfo之后再调用SQLServerUtil.cpp中的方法通过注册表中的内容得到相关的参数。利用实例名称连接数据库，利用DBMS查询数据库的信息。 |
| checkSQLServerDBExistFromClient | 检查该实例下是否有该数据库 |  |
| getSQLServerInstanceVersionFromClient | 获取实例的版本信息根据UUID |  |
| checkSQLServerDBDirExistFromClient | 判断数据库恢复所需的目录是否存在 |  |
| getMaxFreeDiskSpaceFromClient | 获取磁盘最大的剩余空间 |  |
| getOracleInfoFromClient | 获取Oracle的信息 | 根据UUID调用ClientCallBack中getAllOracleInfo |
| getMysqlInfoFromClient | 获取Mysql的信息 |  |

第四章 编译源码

4.1DrcClient和DrcServer的编译步骤

4.1.1源码下载

利用svn->checkout(v2platform/trunk)->将整个v2platform下载下来。

4.1.2在本地配置工程运行的环境

2.1安装vs2010

2.2安装qt-opensource-windows-x86-vs2010-4.8.6

Note:这里可以安装在任意路径下,在QT\bin\qtvars.bat会自动配置环境变量。

2.3安装Ice-3.5.1-6.msi。安装路径为：D:\Program Files (x86)\下

4.1.3window端

在windows端，在client的.sln中进入这个项目。

3.1首先对DRCPlatformLib右键点击“重新生成”，这个时候会在client/Release生成DRCPlatformLib.lib。

3.2之后对DRCClient右键点击“重新生成”，这个时候会在client/Release生成DRCClient.exe,这个就是windows下的客户端。

4.1.4linux端

4.1利用sftp协议，将v2platform中的client，drcLib，server上传到服务器上，注意在上传的时候需要将vs关闭。

4.2在drcLib/DRCPlatformLib/DRCPlatformLib 输入make会生成DRCPlatform.a这个库文件。

4.3 在client/DRCCLient 输入make会生成drc\_client即linux端下的客户端。

4.4在server先“make create\_src”，输入make会生成drc\_server即服务端。

4.2需要哪些组件/依赖才能编译

DRCPlatformLib

4.3编译的操作系统环境

Linux 与 windows

第五章 总结

5.1整体结构的数据流图



5.2客户端内容一览



5.3服务端一览

