# 红鸟平台开发指南



文档名称

红鸟平台客户端开发指南

红鸟网络科技有限公司

日期

2018-03-141410

开发部

针对平台

红鸟平台

版权

红鸟网络科技

目录

[一.客户端与服务器的关系 **22**](#_TOC_250028)

1.[1 红鸟服务器群落 22](#_TOC_250027)

1.[2 客户端与 A 服务器 23](#_TOC_250026)

1.[3 客户端与 M 服务器 24](#_TOC_250025)

1.[4 游戏房间与 G 服务器 25](#_TOC_250024)

1.[5 红鸟的网络消息包的结构 26](#_TOC_250023)

[二.平台服务端结构组成概述 **31**](#_TOC_250016)

[2.1 服务端主要组成模块及其功能 31](#_TOC_250015)

[三.平台服务端程序设计思路 **32**](#_TOC_250014)

3.1[服务端基本框架 32](#_TOC_250013)

3.2网络模块：CTCPSocketManage. 33

3.3逻辑模块基类：CBaseMainManage. 33

3.4[数据库模块基类：CAFCDataBaseManage 33](#_TOC_250012)

3.5[Aserver框架 34](#_TOC_250011)

3.6[MServer 框架 35](#_TOC_250010)

3.7[GameManage 框架 35](#_TOC_250009)

3.8[服务端的基本数据流程 36](#_TOC_250008)

[四. GServer与游戏服务端的调用 **38**](#_TOC_250007)

4.1CGameDesk 提供给游戏服务端的接口 38

4.2[GServer装载游戏服务端 dll的方法 39](#_TOC_250006)

[五.服务端相关的数据库 **39**](#_TOC_250005)

5.1[AServer服务端使用的主要表 39](#_TOC_250004)

5.2[AServer服务端使用的主要存储过程 39](#_TOC_250003)

5.3[GServer服务端使用的主要表 40](#_TOC_250002)

5.4[GServer服务端使用的主要存储过程 40](#_TOC_250001)

5.5GameManage服务端使用的主要表 40

5.6[GameManage服务端使用的主要存储过程 40](#_TOC_250000)

## 一.客户端与服务器的关系

### 1.1 红鸟服务器群落

* 1. 红鸟服务器支持分布服务，通过新建服务器可达到扩容的目的，形成红鸟服务器群落。无论服务器群落如何大，服务端的程序主要分为三类，中心服务器(A 服务器,cocos客户端未使用)、主服务器(M 服务器)、及游戏服务器(G 服务器)。三个服务器各自

工作的侧重点。A 服务器提供接入；M 服务器平台基础服务(如登陆，游

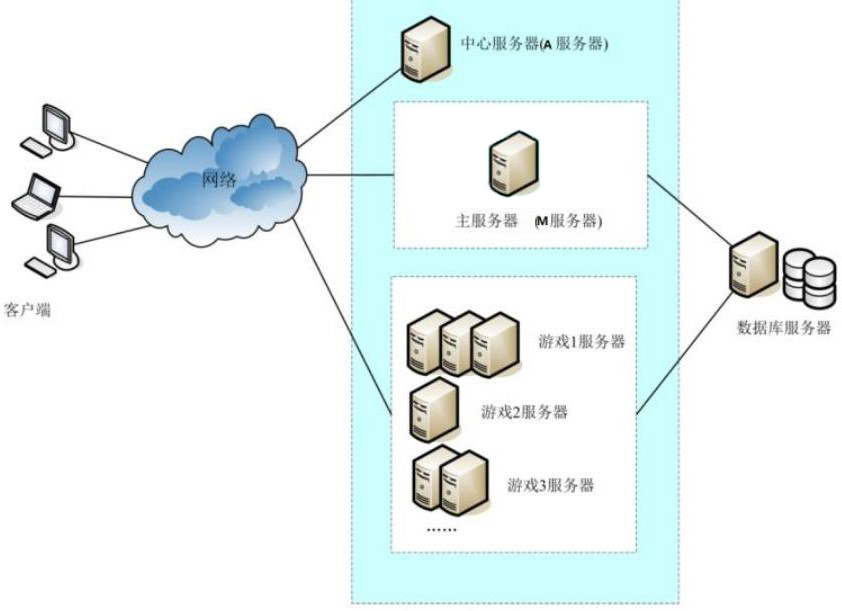
戏列表，好友系统等)；G 服务器负责游戏逻辑与房间坐位服务。

* 1. 红鸟平台的正常运营，需要 A、M、G 服务器同时运行。而平台的基本运行，只需要 A、M 服务器即可。G 服务器属于增值扩充，在乎运营商购买了多少款游戏，没有 G 服务平台也运行，只是不能进行游戏。多个 G 服务器，但只有上一个 A 服务器和 M 服务器。仅有cocos客户端时，不需要开启A。
  2. M、G 服务器一方面对客户端的请求进行处理，别一方面需要与数据库交流。客户端的请求通过 M、G 服务器，M、G 进行逻辑处理，必要使用数据时

M、G 从数据库中读写数据。目前 A 服务器不与数据库连接，A 服务器工作任务

比较简单，也没有需要数据库的内容。

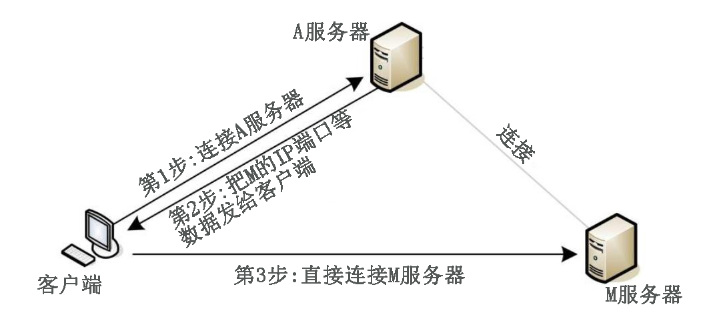
* 1. 总体来说，A、M、G 服务器群落如下图所示。



大厅界面各种窗口组装关系

### 1.2 客户端与 A 服务器（cocos客户端跳过此步骤）

* 1. 客户端启动首先接入 A 服务器，大厅的基础服务全在 M 与 G 服务器中， 客户端启动的最终目的是连接 M 服务器。AServer读取配置文件中设置的MServer地址及端口返回给客户端，客户端根据信息即可连接 M 服务器， 然后客户端可与 A 服务器断开接连。所以 A 服务器中客户端插入服务器的第一道门，其工作任务是十分简单的。工作步骤如下图所示：



### 1.3 客户端与 M 服务器

### 1.4 游戏房间与 G 服务器

* 1. 用户登陆大厅后，M 服务器后大厅发送游戏列表信息，这个游戏列表只包含游戏 ID、游戏名及游戏类型的信息。客户端利用这个数据组织起大厅左边的游戏列表界面。这个时间客户端程序只有游戏的信息并没有游戏对应的房间信息。那么游戏的房间列表是从何而来？答案是展开游戏列表，如下图。用展开击游戏列表的树形控件的游戏名时，客户端才向 M 服务器请对就游戏的房间列表。当客户端收到这个游戏房间列表数据，树形控件该游戏下就能显示出若干房间。
  2. 游戏房间列表数据包括房间名，房间 ID，及房间服务器(即游戏服务器或称 G 服务器)的 IP 与端口。这个数据包的房间的 IP 与端口尤其重要，关系到客户端如何连接 G 服务器，如何进入游戏房间。每个游戏房间的 IP 与端口数据绑定到房间列表中。当用户双击树形控件的游戏房间时，客户端取出该房间对应的 G

服务器的 IP 与端口，创建一个新的 socket 连接之。同样，如果平台支持可同时

进多个房间。那么每进一个游戏房间就多建一个 socket 连接对应的 G 服务器。所以整个平台的与服务器普通运行时的网络连接数为 1 个恒连接 M 服务器的

socket 加上若干个连接 G 服务器的 socket。

1. 连接 G 服务器的 socke，随退出房间而关闭。
2. 如果一个游戏曾经加载过房间列表，则在用户退出前，再次点击游戏树形列表中的游戏名字，则不予以向 M 请求。那么说，游戏房间列表再每次登陆后只获取一次。所以如果 G 的房间有增减及修改，A 与所有的 M 服务器都需要重启，客户端也需要重启，否则客户端获取的房间列表仍然是增减修改前的数据。

### 1.5 红鸟的网络消息包的结构

###### 大厅消息

* MDM\_GP\_NETSIGNAL 网络消息
* MDM\_GP\_BANK (主 ID) 银行消息 ASS\_GP\_BANK\_TRANS\_RECORD (辅助 ID)

转帐记录

* MDM\_GP\_LOGON (主 ID) 登录信息 ASS\_GP\_LOGON\_LOCK\_VALID (辅助 ID)

机器被锁定

* MDM\_GP\_USERINFO (主 ID) 用户资料管理 ASS\_GP\_USERINFO\_ACCEPT 接受用户消息
* MDM\_GP\_SYSMSG (主 ID) 系统消息 ASS\_GP\_NEWS\_SYSMSG 新闻
* MDM\_GP\_PROP 用户道具消息 ASS\_PROP\_BUY 购买道具

对应银行，登录，用户信息，系统消息，道具功能模块。

###### 房间消息

大至按用户动作消息，游戏消息，框架消息等进行分类网络消息

* MDM\_GP\_NETSIGNAL 网络信号消息
* MDM\_GR\_MONEY 钱柜划账 扣钱 ASS\_GR\_TRANSFER\_RECORD 转帐记录
* MDM\_GR\_LOGON 登录 ASS\_GR\_LOGON\_SUCCESS 登录成功
* MDM\_GR\_USER\_LIST 用户列表 ASS\_GR\_ONLINE\_USER 在线用户
* MDM\_GR\_USER\_ACTION 用户动作消息 ASS\_GR\_USER\_UP 用户起来

### 二.平台服务端结构组成概述

###### 2.1服务端主要组成模块及其功能

* + 1. AServer.exe:中心服务器， M 服务器管理，大厅版本升级，cocos客户端未使用
    2. MServer.exe：登录服务器，登录，游戏列表，好友，银行，道具，邮件

等大厅功能

* + 1. GServer.exe：房间管理服务器，加载房间组件；
    2. AFCComDllEx.dll：公用库，数据库，网络等公用模块；
    3. GameManage.dll：房间组件，游戏服务端组件管理，服务器主要负责房间登录，桌子列表;
    4. 用户列表，银行，道具（使用），房间内的聊天，用户房间内动作包括（坐

下，起来，同意等），游戏流程（开始，结束，结算等），排队机房间的除外。比赛场的除外等.

7 CfgEngine.dll：\*.bcf文件操作库

8 HNlibSqlEngine.dll：数据库操作封装库

9 AFCComEx模块的功能

10 MServer模块的功能

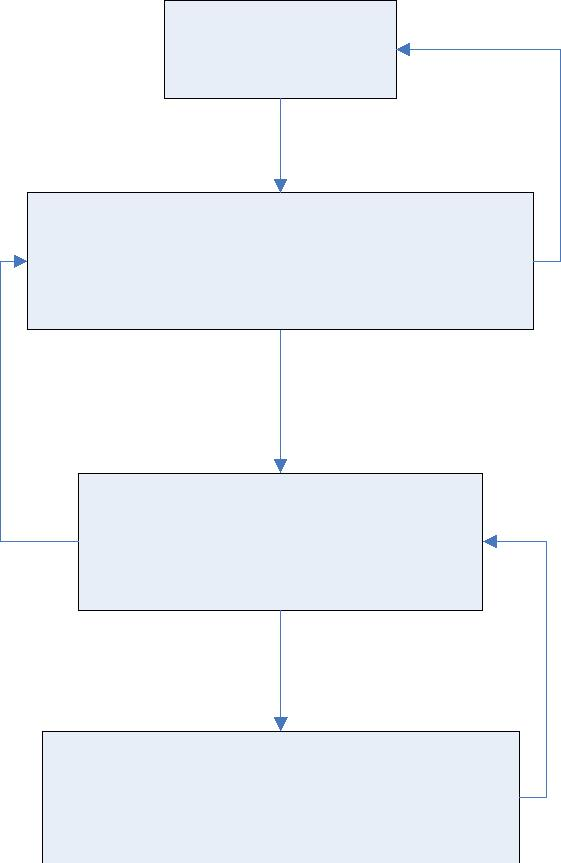
11 GameManage模块的功能

12 GServer模块的功能

13 AServer模块的功能

### 三.平台服务端程序设计思路

###### 3.1服务端基本框架



客户端数据

网络模块（可视为单独线IOCP)

基类：CTCPSocketManage

逻辑模块

基类：CBaseMainManage

数据库模块

基类：CAFCDataBaseManage

如图所示：

3.2网络模块：**CTCPSocketManage**

* + 1. 负责接收来至客户端的消息，使用完成端口实现（实际上就是利用了系统实现的可以处理异步 IO 的数据队列） 详细的过程就是系统会把收到的 TCP 消息，放到指定的地方。读网络数据的线程在有完整的 TCP包时会收到系统的通知， 去指定的地方取数据合并给完整的数据包后， 再调用逻辑模块的

OnSocketRead将数据包传递给逻辑模块来处理。

* + 1. 网络模块最主要的类为：CTCPSocket，CTCPSocketManage等

3.3逻辑模块基类：**CBaseMainManage**

1. 有 客 户 端 消 息 需 要 处 理 时 ， 网 络 模 块 首 先 调 用 逻 辑 模 块 的OnSocketRead，进行相应的处理（包括判断包的合法性，还有一系列的逻辑处理），如果不需要进行数据库进行操作，就将处理结果直接通过网络模块发送给客户端。否则就将数据库操作指令添加到数据库操作队列中去。当数据库模块的读取线程调用到相应指令后，就会根据指令的内容去操作存储过程，当存储过程的值返回后调用逻辑模块 OnDataBaseResult，逻辑模块再将存储过程返回的结果进行进一步的处理，再将最终结果通过网络模块发送给客户端。
2. 逻辑模块最主要的类为：CBaseMainManage，CBaseMainManageZ等

###### 3.4数据库模块基类：CAFCDataBaseManage

* 1. 单独一个线程，PushLine接口提供给逻辑模块添加指令加队列中，当队列中有数据时，根据指令内容执行相应的存储过程，并获取存储过程执行完毕后返回的结果。然后调用逻辑模块 OnDataBaseResult函数将结果传递给逻辑模块来处理。
  2. 数据库模块最主要的类为：CAFCDataBaseManage，CDataBaseHandle，

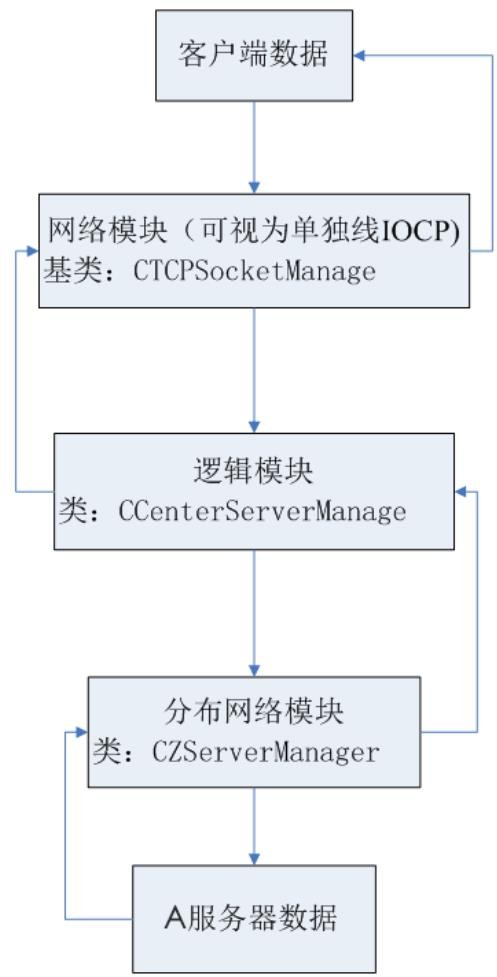
CDataLine等

* 1. 每个服务器的实现基本分为两步：
  2. 首先是要继承 CBaseMainManage 或者 CBaseMainManageForZ，最主要的

目的就是实现其中的 OnSocketRead函数，以及 OnDataBaseResult函数

* 1. 继承 CAFCDataBaseManage，主要是实现 HandleDataBase函数

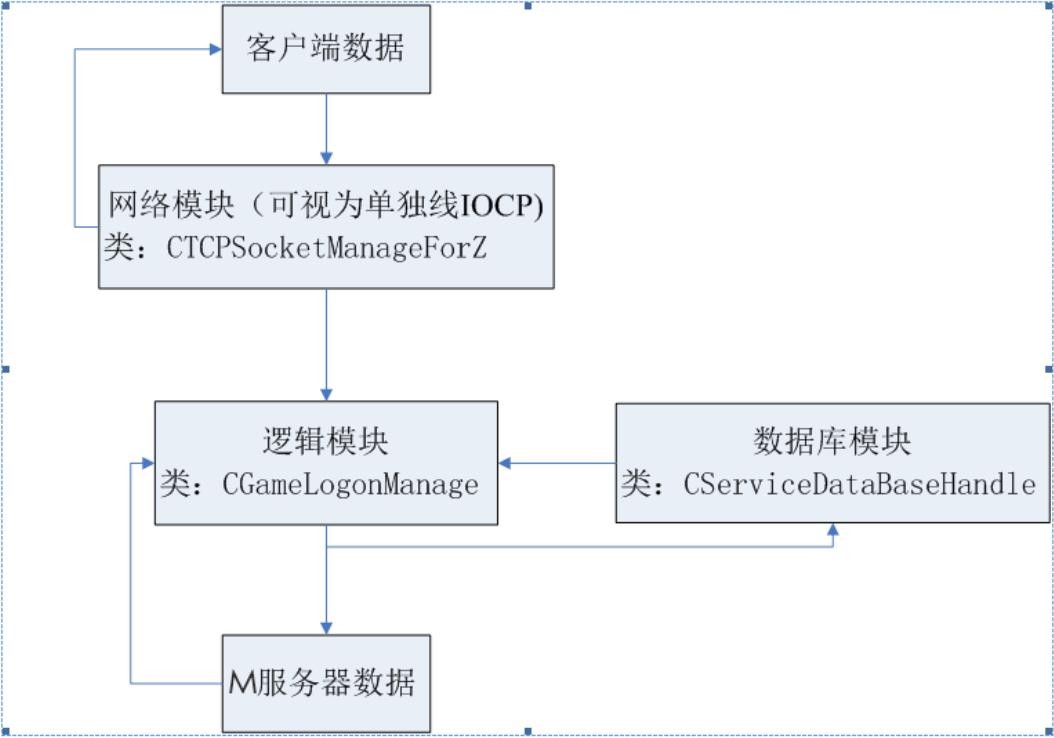
###### Aserver框架



* + 1. A 服务器的实现为如上图所示：
    2. CCenterServerManage类继承 CBaseMainManage
    3. 因为 A 服务器并不需要处理数据库所以不用继承 CAFCDataBaseManage

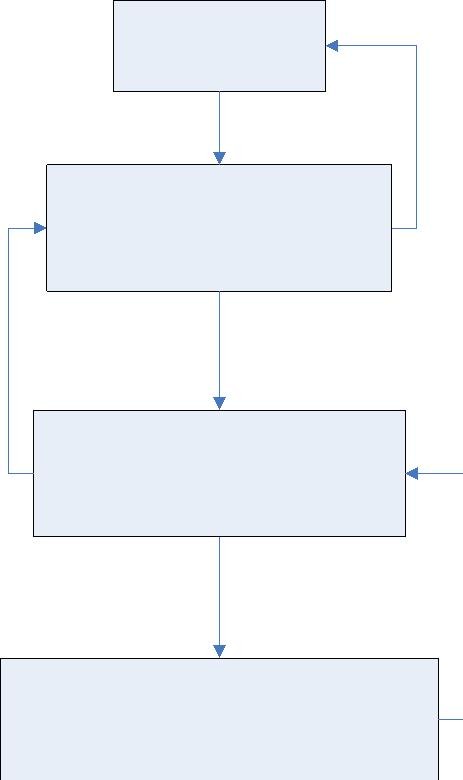
来重写 HandleDataBase函数

* 1. **MServer** 框架



* + 1. M 服务器的实现为如上图所示：
    2. CGameLogonManage类继承 CBaseMainManage
    3. CServiceDataBaseHandle类继承 CAFCDataBaseManage

##### GameManage 框架



客户端数据

网络模块

类：CTCPSocketManage

逻辑模块

类：CGameMainManage

数据库模块

类：CGameDataBaseHandle

* + 1. G 服务器的实现为如上图所示：
    2. CGameMainManage类继承 CBaseMainManage
    3. CGameDataBaseHandle类继承 CAFCDataBaseManage

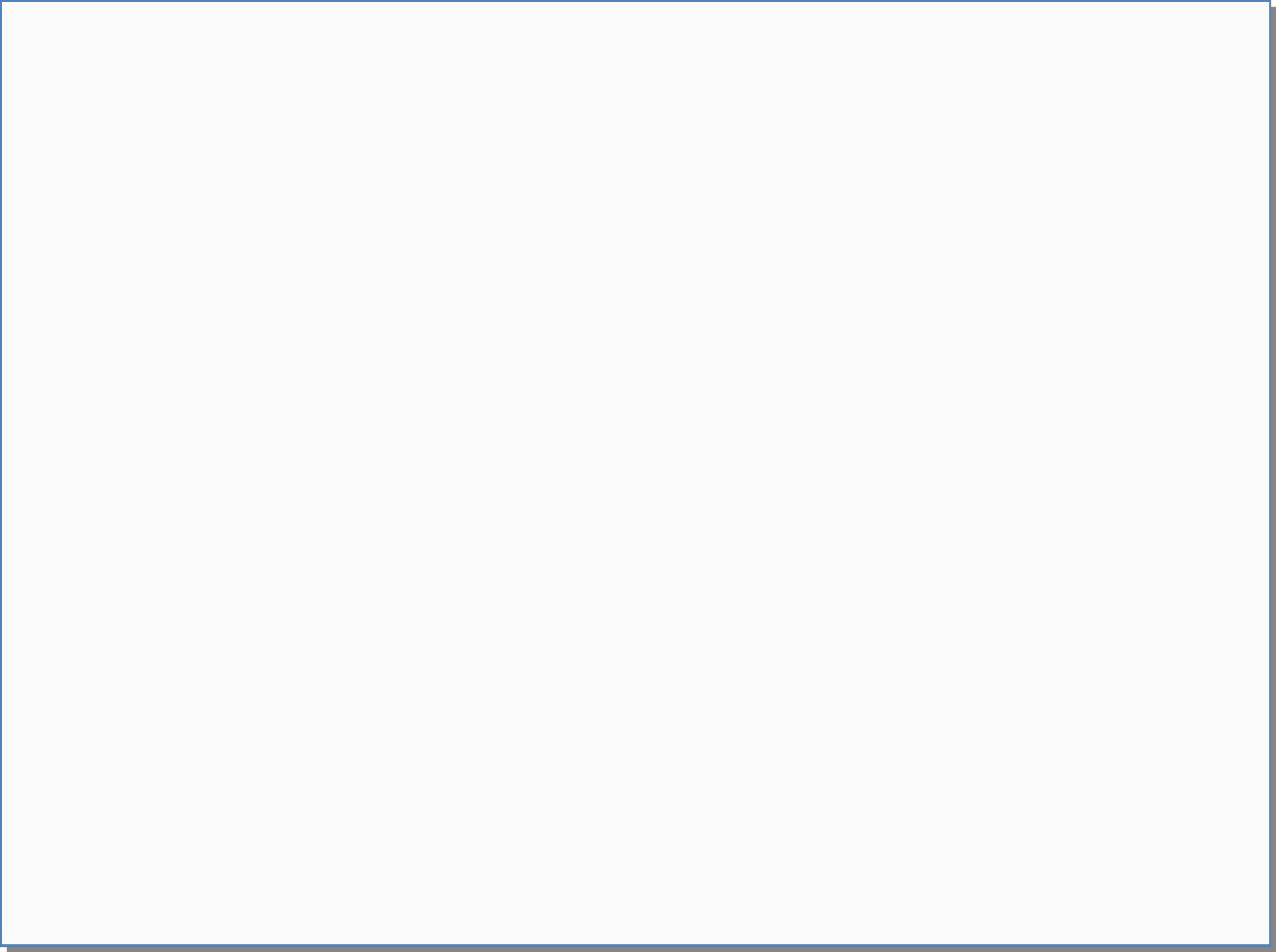
###### 3. 8 服务端的基本数据流程

服务端重要的类及其关系：



### 四. GServer与游戏服务端的调用

* 1. **CGameDesk** 提供给游戏服务端的接口



//是否可以开始游戏

virtual bool CanBeginGame();

//是否可以断线

virtual bool CanNetCut(BYTE bDeskStation);

//用户坐到游戏桌

virtual BYTE UserSitDesk(MSG\_GR\_S\_UserSit \* pUserSit, CGameUserInfo \* pUserInfo);

//用户离开游戏桌

virtual BYTE UserLeftDesk(BYTE bDeskStation, CGameUserInfo \* pUserInfo);

//旁观用户坐到游戏桌

virtual BYTE WatchUserSitDesk(MSG\_GR\_S\_UserSit \* pUserSit, CGameUserInfo \* pUserInfo);

//旁观用户离开游戏桌

virtual BYTE WatchUserLeftDesk(BYTE bDeskStation, CGameUserInfo \* pUserInfo);

//用户同意开始

virtual bool UserAgreeGame(BYTE bDeskStation);

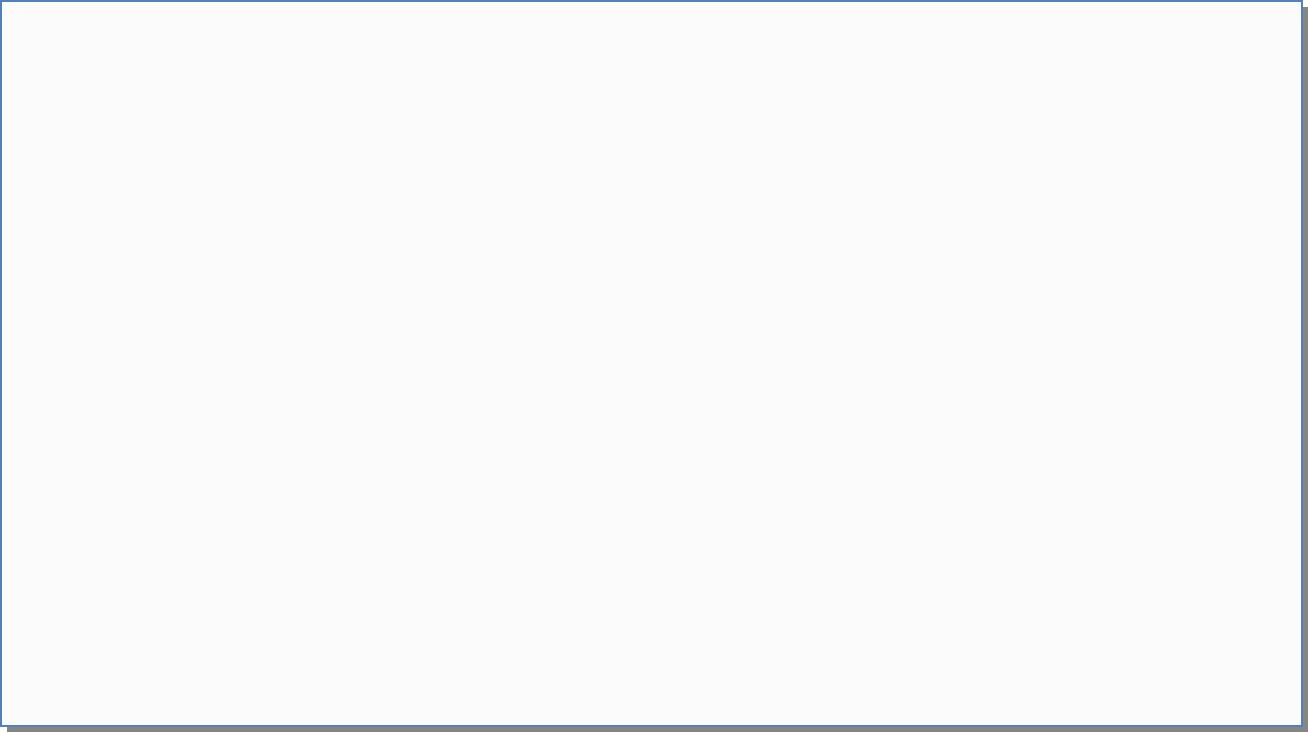
//框架消息处理函数

virtual bool HandleFrameMessage(BYTE bDeskStation, NetMessageHead \* pNetHead, void \* pData, UINT uSize, UINT uSocketID, bool bWatchUser);

//旁观参数变化

virtual bool OnWatchSetChange(BYTE bDeskStation, long int dwTargetUserID, bool bEnableWatch)

###### GServer装载游戏服务端 dll的方法



pComRoomInfo->hDllInstance=AfxLoadLibrary(pComRoomInfo->ServiceInfo.szDLLFileName); if (pComRoomInfo->hDllInstance==NULL)

return false;//throw TEXT("组件装载失败");

//获取管理接口

CreateServiceInterface \* pCreateServiceInterface=(CreateServiceInterface \*) GetProcAddress(pComRoomInfo->hDllInstance,TEXT("CreateServiceInterface"));

if (pCreateServiceInterface==NULL)

return false;//throw TEXT("非法组件");

pComRoomInfo->pIManageService=pCreateServiceInterface(DEV\_LIB\_VER); if (pComRoomInfo->pIManageService==NULL)

return false;//throw TEXT("组件管理接口获取失败，可能是资源不足");

GetServiceInfo \* pGetServiceInfo=(GetServiceInfo\*)

GetProcAddress(pComRoomInfo->hDllInstance,TEXT("GetServiceInfo"));

五.服务端相关的数据库

* 1. AServer服务端使用的主要表
     1. TUserProp
     2. TPropDefine
     3. TUsers
     4. TUserInfo
     5. TZLoginRecord
     6. TGameRoomInfo

###### 5.2 AServer服务端使用的主要存储过程

SP\_UserLoginPlace

SP\_GetOnlineRoom

###### 5.3 GServer服务端使用的主要表

TGameRoomBase

TGameRoomInfo

###### GServer服务端使用的主要存储过程

SP\_GetRoomList

###### GameManage服务端使用的主要表

TGameRoomBase

TGameRoomInfo

TContestInfo

TContestUserRecord

TUsers

TUserInfo

TWLoginRecord

###### GameManage服务端使用的主要存储过程

SP\_UserLoginRoom

SP\_UserInfoUpdateAll