源码项目分析

# 三大神器

每一个神器都会有一个启动服务startService 和一个停止服务 stopService；

首先，每一个神器都会拥有各种引擎实例（引擎helper）和钩子实例,不同的神器会拥有不同的引擎实例和钩子实例。钩子实例中也会拥有各种引擎的指针

每个神器在startService中，首先得到引擎钩子实例，然后将神器中的引擎为参数设置钩子实例中的引擎参数。让神器和钩子实例拥有同样的引擎

## 中心服务器

### 中心服务类CCenterService结构

//组件变量

protected:

CDataBaseSink m\_DataBaseEngineSink; //数据库钩子

CAttemperEngineSink m\_AttemperEngineSink; //调度钩子

//内核组件

protected:

CAttemperEngineHelper m\_AttemperEngine; //调度引擎

CDataBaseEngineHelper m\_DataBaseEngine; //数据引擎

CTCPNetworkEngineHelper m\_TCPNetworkEngine; //网络引擎

* 三个引擎：**调度引擎、数据库引擎、网络引擎**；

调度引擎 CAttemperEngineHelper m\_AttemperEngine;

数据库引擎 CDataBaseEngineHelper m\_DataBaseEngine;

网络引擎 CTCPNetworkEngineHelper m\_TCPNetworkEngine;

数据库引擎钩子：DataBaseSink.h（中心服务器项目实现）

调度引擎钩子：AttemperEngineSink.h（中心服务器项目实现）

网络引擎 CTCPNetworkEngine（在内核引擎的网络引擎中实现）

数据库钩子：中心服务器中，未实现DataBaseSink.h描述的方法

* 两个钩子： **数据钩子、调度钩子**

数据库钩子 CDataBaseSink m\_DataBaseEngineSink;

调度钩子 CAttemperEngineSink m\_AttemperEngineSink;

### 启动流程

中心服务器启动从以下方法中开始

//开始服务bool CCenterService::StartService() ，定义在CenterService.h文件中

1. 检验服务运行状态
2. 加载参数
3. 设置钩子外挂
4. 配置网络端口以及最大连接数
5. 启动三大引擎，根据设置的钩子外挂去执行相应的功能，如启动、停止等，等待事件处理

### 中心服务器引擎钩子模块分析

#### 调度引擎钩子

调度引擎钩子模块：三大引擎接口、服务列表

//接口变量

protected:

ITimerEngine \* m\_pITimerEngine; //时间引擎

IDataBaseEngine \* m\_pIDataBaseEngine; //数据引擎

ITCPNetworkEngine \* m\_pITCPNetworkEngine; //网络引擎

//组件定义

protected:

CServerListCenterHelper m\_ServerListCenter;

CServerListCenterHelper 定义在“列表服务”项目下，主要定义了，关于游戏类型，房间、游戏种类，上线人数等操作。

从内核调度引擎执行的时候，会用到登录服务器结构中的引擎钩子

钩子引擎的设置是根据中心服务器的引擎helper对象的getInstance()方法来设置

#### 数据库引擎钩子

## 登录服务器

### 登录服务类CLogonService结构

//核心结构

protected:

CDataBaseSink m\_DataBaseEngineSink; //数据钩子

CAttemperEngineSink m\_AttemperEngineSink; //调度钩子

//内核组件

protected:

CTimerEngineHelper m\_TimerEngine; //时间引擎

CAttemperEngineHelper m\_AttemperEngine; //调度引擎

CDataBaseEngineHelper m\_DataBaseEngine; //数据引擎

CTCPNetworkEngineHelper m\_TCPNetworkEngine; //网络引擎

CTCPSocketServiceHelper m\_TCPSocketCorrespond; //协调服务

* 两个钩子： **数据库钩子、调度钩子**

CDataBaseSink m\_DataBaseEngineSink; //数据库钩子

CAttemperEngineSink m\_AttemperEngineSink; //调度钩子

* 四大引擎

CTimerEngineHelper m\_TimerEngine; //时间引擎

CAttemperEngineHelper m\_AttemperEngine; //调度引擎

CDataBaseEngineHelper m\_DataBaseEngine; //数据库引擎

CTCPNetworkEngineHelper m\_TCPNetworkEngine; //网络引擎

此四大引擎都是在内核中实现

* 一个服务

CTCPSocketServiceHelper m\_TCPSocketCorrespond; //协调服务

此服务在内核中实现，与网络引擎同目录下

### 启动流程

登录服务器启动从以下方法中开始

//开始服务bool CLogonService::StartService() ，定义在CenterService.h文件中

1. 检验服务运行状态
2. 加载参数
3. 设置绑定钩子外挂
4. 启动三大引擎

时间引擎、网络引擎、调度引擎、数据库引擎

调用内核引擎实现启动；

1. 启动一个服务

网络服务，调用内核引擎实现启动；

### 登录服务器引擎钩子模块分析

#### 调度引擎钩子

调度引擎钩子模块：三大引擎接口、服务器列表、一个服务

//辅助组件

protected:

CServerList m\_ServerList; //服务器列表

//接口变量

protected:

ITimerEngine \* m\_pITimerEngine; //时间引擎

IDataBaseEngine \* m\_pIDataBaseEngine; //数据引擎

ITCPNetworkEngine \* m\_pITCPNetworkEngine; //网络引擎

ITCPSocketService \* m\_pITCPSocketCorrespond; //协调服务

从内核调度引擎执行的时候，会用到登录服务器结构中的引擎钩子

钩子引擎的设置是根据登录服务器的引擎helper对象的getInstance()方法来设置

#### 数据库引擎钩子

## 服务装载器

服务装载器仅仅是一个外壳，核心部分是“游戏服务”工程项目，它才是服务装载器的心脏。

### 游戏服务类CGameService

class CGameService : public IGameService，核心结构如下

protected:

CDataBaseSink m\_DataBaseEngineSink; //数据库钩子

CAttemperEngineSink m\_AttemperEngineSink; //调度钩子

//内核组件

protected:

CTimerEngineHelper m\_TimerEngine; //时间引擎

CAttemperEngineHelper m\_AttemperEngine; //调度引擎

CDataBaseEngineHelper m\_DataBaseEngine; //数据引擎

CTCPNetworkEngineHelper m\_TCPNetworkEngine; //网络引擎

CTCPSocketServiceHelper m\_TCPSocketCorrespond; //协调服务

* 两个钩子： **数据钩子、调度钩子**

数据库钩子 CDataBaseSink m\_DataBaseEngineSink;

调度钩子 CAttemperEngineSink m\_AttemperEngineSink;

* 四大引擎

CTimerEngineHelper m\_TimerEngine; //时间引擎

CAttemperEngineHelper m\_AttemperEngine; //调度引擎

CDataBaseEngineHelper m\_DataBaseEngine; //数据库引擎

CTCPNetworkEngineHelper m\_TCPNetworkEngine; //网络引擎

* 一个服务

CTCPSocketServiceHelper m\_TCPSocketCorrespond; //协调服务

此服务在内核中实现，与网络引擎同目录下

### 启动流程

装载服务器调用“游戏服务”组件，启动从以下方法中开始

//开始服务bool CGameService::StartService() ，定义在CenterService.h文件中

1. 检验服务运行状态
2. 创建引擎实例
3. 加载游戏服务模块组件
4. 设置绑定钩子外挂
5. 启动四大引擎

时间引擎、网络引擎、调度引擎、数据库引擎

调用内核引擎实现启动；

1. 启动一个服务

网络服务，调用内核引擎实现启动；

### 游戏服务引擎钩子模块

#### 调度引擎钩子

//接口变量

protected:

ITimerEngine \* m\_pITimerEngine; //时间引擎

IDataBaseEngine \* m\_pIDataBaseEngine; //数据引擎

IAttemperEngine \* m\_pIAttemperEngine; //调度引擎

ITCPNetworkEngine \* m\_pITCPNetworkEngine; //网络引擎

ITCPSocketService \* m\_pITCPSocketCorrespond; //协调服务

从内核调度引擎执行的时候，会用到登录服务器结构中的引擎钩子

钩子引擎的设置是根据登录服务器的引擎helper对象的getInstance()方法来设置

#### 数据库引擎钩子

# 内核调度引擎

## 调度管理类结构

class CAttemperEngine : public IAttemperEngine, public IQueueServiceSink,

public IDataBaseEngineEvent, public ITCPNetworkEngineEvent

{

//主要核心变量

protected:

bool m\_bService; //运行标志

CQueueService m\_QueueService; //队列对象

CCriticalSection m\_CriticalSection; //同步锁定

//接口变量

protected:

ITCPSocketService \* m\_pITCPSocketService; //网络引擎

ITCPNetworkEngine \* m\_pTCPNetworkEngine; //网络引擎

IAttemperEngineSink \* m\_pIAttemperEngineSink; //挂接接口

//机器人链接维护

protected:

CArrayTemplate<DWORD> m\_AndroidUserSocketIDArray; //机器人连接

## 调度引擎启动过程

调度引擎启动，获取内核调动引擎接口，从内核调动引擎开始执行（服务器组件-> 内核引擎CAttemperEngine::StartService（））；

从CAttemperEngine::StartService() 开始：

1. 判断状态，是否启动；
2. 检查调度引擎外挂钩子设置是否正常，不正常返回；
3. 设置调度引擎与队列服务绑定，设置队列服务外挂钩子；
4. 启动调度引擎外挂钩子；
5. 启动内核引擎下的队列服务；
6. 清空机器人，启动完成，等待事件处理。

## 数据队列接口处理

void \_\_cdecl CAttemperEngine::OnQueueServiceSink(WORD wIdentifier, void \* pBuffer, WORD wDataSize)

队列服务CQueueService 中设置了队列服务钩子，由此钩子产生回调，在回调中根据wIdentifier标识不同的类型进行处理；wIdentifier有以下几种类型

//事件定义

时间事件 #define EVENT\_TIMER 0x0001

控制事件 #define EVENT\_CONTROL 0x0002 数据库事件 #define EVENT\_DATABASE 0x0003

读取事件 #define EVENT\_TCP\_SOCKET\_READ 0x0004

关闭事件 #define EVENT\_TCP\_SOCKET\_CLOSE 0x0005

连接事件 #define EVENT\_TCP\_SOCKET\_CONNECT 0x0006 应答事件 #define EVENT\_TCP\_NETWORK\_ACCEPT 0x0007

读取事件 #define EVENT\_TCP\_NETWORK\_READ 0x0008 关闭事件 #define EVENT\_TCP\_NETWORK\_CLOSE 0x0009

对不属于上面类型的其他定义，丢到调度引擎的钩子实现中去处理，如下

m\_pIAttemperEngineSink->OnEventAttemperData(wIdentifier, pBuffer, wDataSize);

实现钩子的有中心服务器、登录服务器、游戏服务三种，这三个钩子中，均没有实现OnEventAttemperData的处理，也就意味着不会处理不识别的类型。这里只留作将来扩展使用。

下面分别描述不同类型的处理过程

### EVENT\_TIMER 时间事件

直接调用钩子方法中实现的OnEventTimer方法

m\_pIAttemperEngineSink->OnEventTimer(pTimerEvent->dwTimerID, pTimerEvent->dwBindParameter);，有以下三种钩子实现

#### 中心服务器

未实现OnEventTimer的处理

#### 游戏服务

实现如下类型处理，具体的处理请见游戏服务章节的详细分析

case IDI\_CONNECT\_CENTER\_SERVER: //连接中心服务器

case IDI\_UPDATE\_SERVER\_LIST: //更新列表

case IDI\_UPDATE\_ONLINE\_COUNT: //更新人数

case IDI\_SEND\_SYSTEM\_MESSAGE: //系统消息

case IDI\_LOAD\_ANDROID\_INFO: //加载用户

case IDI\_DISTRIBUTE\_ANDROID: //分配用户

#### 登录服务器

实现了两种OnEventTimer事件处理，详细分析请见登录服务器章节

case IDI\_CONNECT\_CENTER\_SERVER: //连接中心服务器定时连接中心服务器，避免连接端口后停止工作。类似看门狗

case IDI\_UPDATE\_SERVER\_LIST: //更新服务器列表

### EVENT\_DATABASE数据库事件

实现了该事件处理钩子的有三种，中心服务器、游戏服务、登录服务器。

#### 中心服务器

未处理

#### 游戏服务

case DBR\_GR\_LOGON\_SUCCESS: //登录成功

case DBR\_GR\_LOGON\_ERROR: //登录失败

case DBR\_GR\_ANDROID\_USER: //机器信息

case DBR\_GR\_USER\_RIGHT\_RESULT: //用户权限

case DBR\_GR\_PROPERTY\_ATTRIBUTE: //加载道具

case DBR\_GR\_FLOWER\_ATTRIBUTE: //加载鲜花

case DBR\_GR\_USER\_PROPERTY: //玩家道具

//case DBR\_GR\_EXCHANGE\_RESULT: //兑换结果

case DBR\_GR\_LOAD\_PROP\_FINISHI: //完成加载

详细处理请看游戏服务章节；

#### 登录服务器。

case DBR\_GP\_LOGON\_ERROR: //登录失败

case DBR\_GP\_LOGON\_SUCCESS: //登录成功

case DBR\_GP\_DOWNLOADFACE\_SUCCESS: //下载成功

case DBR\_GP\_UPLOAD\_FACE\_RESULT: //上传结果

case DBR\_GP\_DELETE\_FACE\_RESULT: //删除结果

case DBR\_GP\_MODIFY\_RESULT: //修改结果

详细分析请见登录服务器章节；

### EVENT\_TCP\_NETWORK\_ACCEPT网络应答事件

调度引擎中，调用钩子的处理方法，见下面：

m\_pIAttemperEngineSink->OnEventTCPNetworkBind(pSocketAcceptEvent->dwClientIP, pSocketAcceptEvent->dwSocketID);

#### 中心服务器

绑定网络

#### 游戏服务

绑定网络

#### 登录服务器。

绑定网络

### EVENT\_TCP\_NETWORK\_READ: 网络读取事件

内核调度引擎实现如下调用，三种钩子实现具体方法

m\_pIAttemperEngineSink->OnEventTCPNetworkRead(pSocketReadEvent->Command, pSocketReadEvent + 1, pSocketReadEvent->wDataSize, pSocketReadEvent->dwSocketID);

#### 中心服务器

case MDM\_CS\_SERVER\_LIST: //列表消息

case MDM\_CS\_SERVER\_MANAGER: //房间管理

具体分析，请见中心服务器章节

#### 游戏服务

case MDM\_GR\_LOGON: //登录消息

case MDM\_GR\_USER: //用户消息

case MDM\_GR\_MANAGER://管理消息

case MDM\_GF\_FRAME: //框架消息

case MDM\_GF\_GAME: //游戏消息

case MDM\_GF\_VIDEO: //视频消息

case MDM\_GF\_PRESENT://礼物消息

case MDM\_GF\_BANK: //银行消息

具体分析，请见游戏服务器章节

#### 登录服务器。

case MDM\_GP\_LOGON: //登录消息

case MDM\_GP\_USER: //用户信息

具体分析，请见登录服务器章节

### EVENT\_TCP\_NETWORK\_CLOSE: 关闭读取事件

调度引擎中，调用钩子的处理方法，见下面：

m\_pIAttemperEngineSink->OnEventTCPNetworkShut(pSocketCloseEvent->dwClientIP, pSocketCloseEvent->dwActiveTime, pSocketCloseEvent->dwSocketID);

#### 中心服务器

见钩子实现文件

#### 游戏服务

见钩子实现文件

#### 登录服务器。

见钩子实现文件

### EVENT\_TCP\_SOCKET\_READ: 网络读取事件

调度引擎中，调用钩子的处理方法，见下面：

m\_pIAttemperEngineSink->OnEventTCPSocketRead(pSocketReadEvent->wServiceID, pSocketReadEvent->Command, pSocketReadEvent + 1, pSocketReadEvent->wDataSize);

#### 中心服务器

未处理该事件

#### 游戏服务

case MDM\_CS\_SERVER\_LIST: //列表消息

具体处理方法见钩子实现文件

#### 登录服务器。

case MDM\_CS\_SERVER\_LIST: //列表消息

具体处理方法见钩子实现文件

### EVENT\_TCP\_SOCKET\_CLOSE: 网络关闭事件

调度引擎中，调用钩子的处理方法，见下面：

m\_pIAttemperEngineSink->OnEventTCPSocketShut(pSocketCloseEvent->wServiceID, pSocketCloseEvent->cbShutReason);

#### 中心服务器

未处理该事件

#### 游戏服务

具体处理方法见钩子实现文件

#### 登录服务器。

具体处理方法见钩子实现文件

### EVENT\_TCP\_SOCKET\_CONNECT: 网络连接

调度引擎中，调用钩子的处理方法，见下面：

m\_pIAttemperEngineSink->OnEventTCPSocketLink(pSocketCloseEvent->wServiceID, pSocketCloseEvent->nErrorCode);

#### 中心服务器

未处理该事件

#### 游戏服务

具体处理方法见钩子实现文件

#### 登录服务器。

具体处理方法见钩子实现文件

### EVENT\_CONTROL 事件控制

调度引擎中，调用钩子的处理方法，见下面：

m\_pIAttemperEngineSink->OnEventControl(wIdentifier, pBuffer, wDataSize);

#### 中心服务器

未处理该事件

#### 游戏服务

未处理该事件

#### 登录服务器。

未处理该事件

# 内核数据库引擎

1. 检查外挂钩子有效性
2. 设置数据队列钩子

启动数据库外挂钩子

# 内核定时器引擎

由内核引擎的时间引擎接口开始执行 CTimerEngine::StartService（）

# 内核网络引擎

由内核引擎的网络引擎接口开始执行 CTCPNetworkEngine::StartService()

# 内核网络服务

由内核引擎的网络引擎接口开始执行 CTCPSocketService::StartService（）