服务器核心模块概要设计

# 1 引言

## 1.1 编写目的

方便二次开发人员理解服务器核心技术

## 1.2 背景

3.3.1 为一般游戏服务器设计的核心模块

3.3.2 本项目的任务开发者:Godzilar

系统使用用户: 游戏服务器二次开发人员

# 2 总体设计

## 2.1 需求规定

### 3.3.1 系统功能

提供网络服务接口,数据库访问接口,定时器接口,并提供调度钩子接口,让二次开发人员调用,更多精力处理游戏逻辑

### 3.3.2 系统性能

网络负载支持 2000 同时在线连接以上

数据库访问支持 100 并发访问

定时器时间精度达 25ms 之内

## 2.2 运行环境

硬件平台:x86 体系结构

软件平台:Window2003 操作系统

## 2.3 开发环境

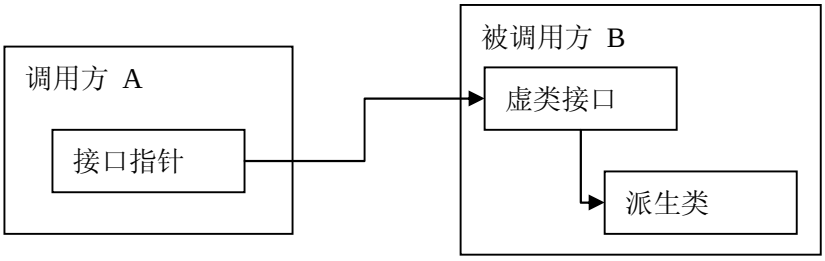
开发工具:Visual C++ 2003

开发语言:C++

## 2.4 基本设计概念和处理流程

### 2.4.1 调用模型

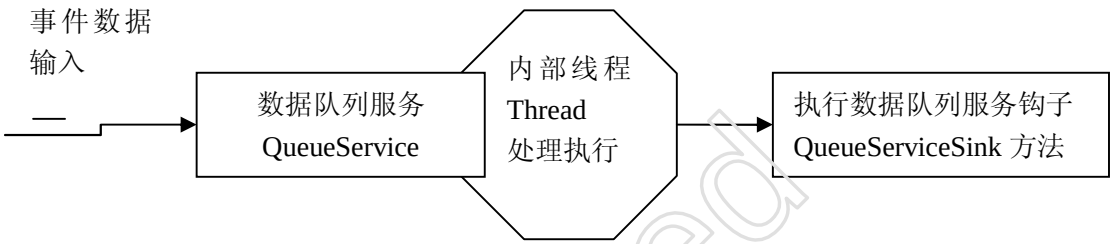
利用面向对象思想多态性,**调用方**保存着**被调用方**的基础接口指针(**一般称呼为 钩子**),调用方直接调用接口指针里面方法,方法具体实现逻辑由该接口的派生类实现.示意图:



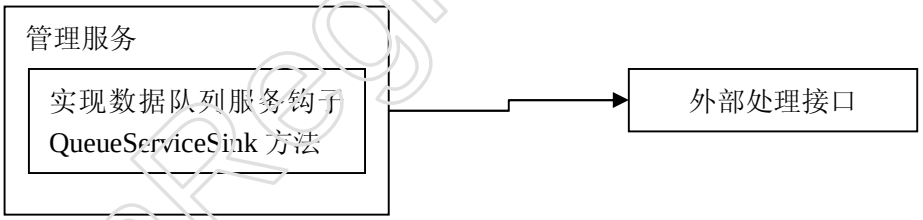
### 2.4.2 基于事件驱动的数据处理模型

为系统功能设计，需要处理网络事件数据 SocketEvent，数据库事件数据 DatabaseEvent，定时器事件数据 TimerEvent 等，为此建立数据队列服务 QueueService,为每一队列建立多个子线程 QueueServiceThread 处理。数据队列服务提供添加事件数据方法 AddToQueue，设置数据队列服务钩子 SetQueueServiceSink，让数据队列服务钩子 QueueServiceSink 做具体逻辑事件数据处理。

示意图:



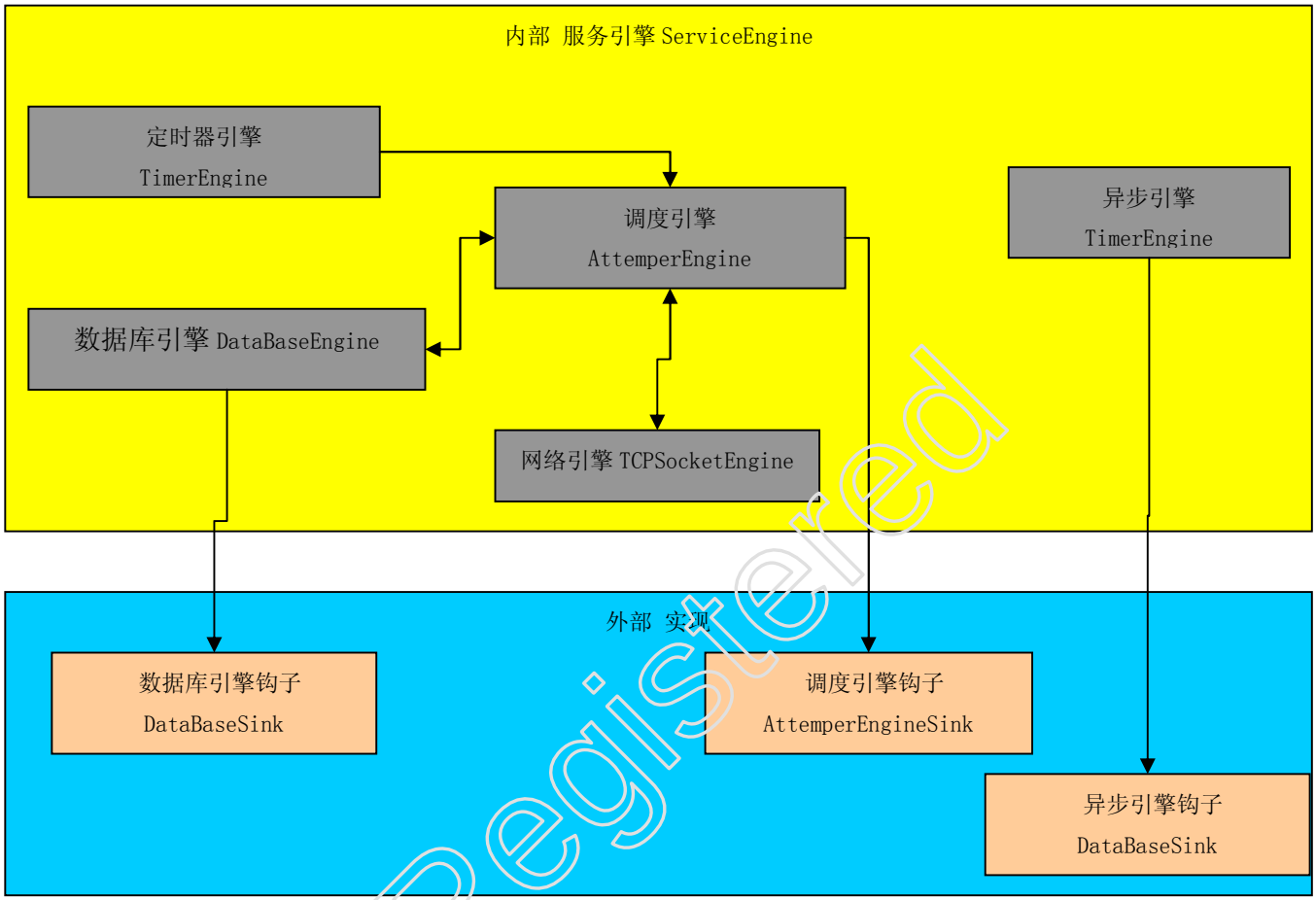

再者，根据网络,数据库等特定功能，构建网络，数据库等管理服务，实现数据队列服务钩子 QueueServiceSink 方法，调度事件数据执行外部处理接口，外部处理接口具体由二次开发用户实现。示意图:




### 2.4.3 事件数据设计

| 名称 | 描述 | 类型 |
|---|---|---|
| 定时器事件 TimerEvent | 根据时间间隔,定期产生 | 自发,被动 |
| 数据库事件 DatabaseEvent | 请求访问数据库 | 用户自定义，主动 |
| 网络应答事件 SocketAcceptEvent | 客户端连接网络服务,成功连接 | 自发，被动 |
| 网络读取事件 SocketReadEvent | 客户端发送数据,网络服务成功读取 | 自发，被动 |
| 网络关闭事件 SocketControlEvent | 客户端失去连接 | 自发，被动 |
| 控制事件 ControlEvent | 用户自定义控制服务 | 用户自定义，主动 |

### 2.4.4 框架体系设计






## 2.5 功能模块划分

| 名称 | 描述 | 开发优先 |
|---|---|---|
| 数据队列服务<br>QueueService | 外部接受输入事件数据，内部实现事件数据链表保存与管理，处理子线程从链表获取事件数据调用数据队列钩子方法处理事件数据 | 1 |
| 网络引擎<br>TCPSocketEngine | 实现数据队列服务钩子方法，管理网络事件数据，内部启动网络服务器监听服务,接收与发送网络数据,并调用网络处理钩子进行逻辑处理 | 2 |
| 数据库引擎<br>DataBaseEngine | 实现数据队列服务钩子方法，管理数据库事件数据，并调用数据库处理钩子进行逻辑处理，还提供数据库访问的帮助类 | 2 |
| 定时器引擎<br>TimerEngine | 实现设置定时,删除定时请求，发生定时事件 | 2 |
| 调度引擎 | 接收，调度与管理网络事件数据，数据库事件数据，定时 | 2 |

| | | |
|---|---|---|
| AttemperEngine | 器事件数据，自定义调度事件数据 | |
| 异步引擎<br>AsynchronismEngine | 设置多个数据处理钩子，异步接收请求并处理 | 2 |
| 服务引擎<br>ServiceEngine | 管理网络引擎，数据库引擎，定时器引擎，调度引擎，异步引擎 | 3 |

# 3 模块详细设计

## 3.1 数据队列服务 QueueService 模块设计

### 3.1.1 基本设计概念

建立一个内存链表，保存事件数据，对外部提供方法往链表添加事件数据并通知线程，同时启动多个处理线程，从数据链表里获取事件数据，执行外部钩子方法进行处理. 线程事件通知采用完成端口技术.

### 3.1.2 类与接口设计

- **接口设计**

| IQueueServiceEngine | | |
|---|---|---|
| 队列引擎接口 | | |
| 模块 | QueueService | |
| 继承 | IQueueServiceEngine → IUnknownEx | |
| 实现 | | |
| 数据成员 | | |
| 名称 | 说明 | |
| | | |
| 方法 | | |
| virtual bool __cdecl StartService(BYTE cbThreadCount) | | |
| 说明 | 开始服务 | |
| 参数 | BYTE cbThreadCount | 处理事件数据的线程数量 |
| 返回 | 类型为 Bool，启动成功返回 true，否则 false | |
| 举例 | | |
| virtual bool __cdecl StopService() | | |
| 说明 | 停止服务 | |
| 参数 | | |
| 返回 | 类型为 Bool，启动成功返回 true，否则 false | |
| 举例 | | |
| virtual bool __cdecl SetQueueServiceSink(IUnknownEx * pIUnknownEx) | | |
| 说明 | 设置接口外部事件处理钩子 | |
| 参数 | IUnknownEx * pIUnknownEx | 外部事件数据处理钩子 |
| 返回 | 类型为 Bool，设置成功返回 true，否则 false | |
| 举例 | | |
| virtual bool __cdecl GetBurthenInfo(tagBurthenInfo & BurthenInfo) | | |
| 说明 | 获取数据队列负荷信息 | |
| 参数 | tagBurthenInfo & BurthenInfo | 负荷信息结构体 |
| 返回 | 类型为 Bool，设置成功返回 true，否则 false | |

| 举例 | |
|---|---|
| | |

| IQueueServiceSink | | |
|---|---|---|
| 队列服务钩子,由外部继层实现 | | |
| 模块 | QueueService | |
| 继承 | IQueueServiceSink → IUnknownEx | |
| 实现 | | |
| 数据成员 | | |
| 名称 | | 说明 |
| | | |
| 方法 | | |
| virtual void __cdecl OnQueueServiceSink(BYTE cbThreadIndex, WORD wIdentifier, void * pBuffer, WORD wDataSize, DWORD dwInsertTime) | | |
| 说明 | 通知回调函数 | |
| | 参数 | BYTE cbThreadIndex 处理事件数据的线程索引<br>WORD wIdentifier 事件数据标记<br>void * pBuffer 数据指针<br>WORD wDataSize 数据大小<br>DWORD dwInsertTime 插入时间值 |
| | 返回 | |
| 举例 | | |
| | | |

- **类设计**

| CQueueServiceThread | | |
|---|---|---|
| 队列处理线程类 | | |
| 模块 | QueueService | |
| 继承 | CQueueServiceThread → CServiceThread | |
| 实现 | | |
| 数据成员 | | |
| 名称 | | 说明 |
| HANDLE m_hCompletionPort | | 完成端口句柄 |
| BYTE m_cbThreadIndex | | 线程索引 |
| BYTE m_cbBuffer[MAX_QUEUE_PACKET] | | 事件接收缓冲 |
| 方法 | | |
| bool InitThread(HANDLE hCompletionPort,BYTE cbThreadIndex) | | |
| 说明 | 配置 | |
| | 参数 | HANDLE hCompletionPort 完成端口句柄<br>BYTE cbThreadIndex 线程索引 |
| | 返回 | 类型为 Bool,设置成功返回 true,否则 false |

| | |
|---|---|
| 举例 | |

| bool UnInitThread() | | |
|---|---|---|
| 说明 | 取消配置 | |
| | 参数 | |
| | 返回 | 类型为 Bool,设置成功返回 true,否则 false |
| 举例 | | |

| virtual bool RepetitionRun() | | |
|---|---|---|
| 说明 | 线程运行函数 | |
| | 参数 | |
| | 返回 | 类型为 Bool |
| 举例 | | |
| | | |

| CQueueService | |
|---|---|
| 队列服务类 | |
| 模块 | QueueService |
| 继承 | CQueueService → IQueueServiceEngine, IQueueService |
| 实现 | IQueueServiceEngine, IQueueService |

| 数据成员 | |
|---|---|
| 名称 | 说明 |
| bool m_bService | 服务标志 |
| IQueueServiceSink * m_pIQueueServiceSink | 回调钩子接口 |
| CThreadLock m_ThreadLock | 线程锁 |
| CDataStorage m_DataStorage | 数据存储 |
| CQueueServiceThread* m_QueueServiceThread | 队列线程 |
| HANDLE* m_hCompletionPort | 完成端口指针 |
| BYTE m_cbThreadCount | 线程处理数目 |

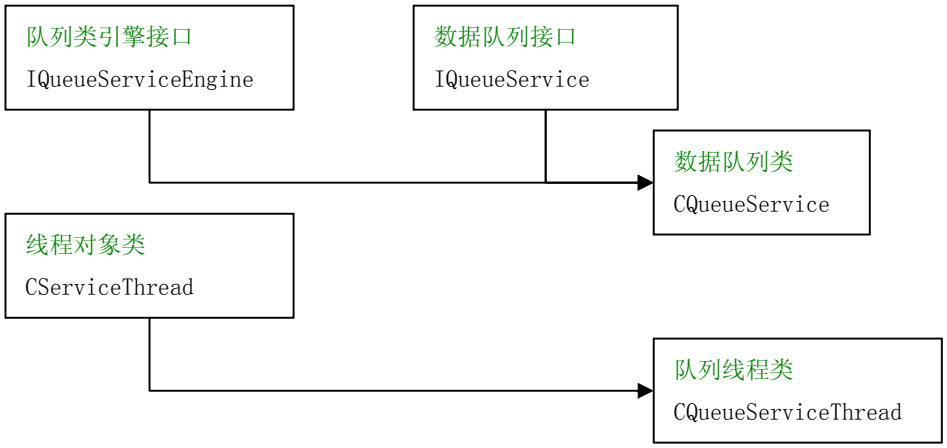
方法

| virtual bool __cdecl AddToQueue(WORD wIdentifier, void * const pBuffer, WORD wDataSize) | | |
|---|---|---|
| 说明 | 添加事件数据 | |
| | 参数 | WORD wIdentifier 事件数据标记<br>void * const pBuffer 事件数据指针<br>WORD wDataSize 事件数据大小 |
| | 返回 | 类型为 Bool,添加成功返回 true,否则 false |
| 举例 | | |

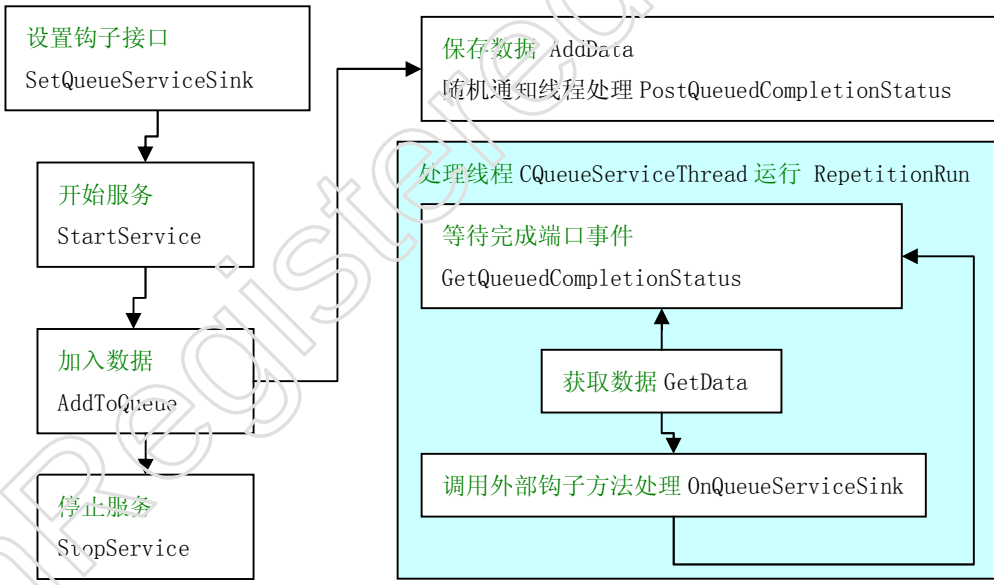
| virtual bool __cdecl StartService(BYTE cbThreadCount) | | |
|---|---|---|
| 说明 | 开始服务 | |
| | 参数 | BYTE cbThreadCount 线程数量 |
| | 返回 | 类型为 Bool,开始成功返回 true,否则 false |
| 举例 | | |

| virtual bool __cdecl StopService() | | |
|---|---|---|
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 类型为 Bool, 停止成功返回 true,否则 false |
| 举例 | | |
| virtual bool __cdecl SetQueueServiceSink(IUnknownEx * pIUnknownEx) | | |
| 说明 | 设置接口外部事件处理钩子 | |
| | 参数 | IUnknownEx * pIUnknownEx 外部事件数据处理钩子 |
| | 返回 | 类型为 Bool,设置成功返回 true,否则 false |
| 举例 | | |
| virtual bool __cdecl GetBurthenInfo(tagBurthenInfo & BurthenInfo) | | |
| 说明 | 获取数据队列负荷信息 | |
| | 参数 | tagBurthenInfo & BurthenInfo 负荷信息结构体 |
| | 返回 | 类型为 Bool,设置成功返回 true,否则 false |
| 举例 | | |
| bool GetData(tagDataHead & DataHead, void * pBuffer, WORD wBufferSize) | | |
| 说明 | 获取事件数据 | |
| | 参数 | tagDataHead & DataHead 事件数据头<br>void * pBuffer 数据指针<br>WORD wBufferSize 数据大小 |
| | 返回 | 类型为 Bool,获取成功返回 true,否则 false |
| 举例 | | |
| void OnQueueServiceThread(BYTE cbThreadIndex, const tagDataHead & DataHead, void * pBuffer, WORD wDataSize) | | |
| 说明 | 事件消息,主要由队列线程调用 | |
| | 参数 | bYTE cbThreadIndex 线程索引<br>tagDataHead & DataHead 事件数据头<br>void * pBuffer 数据指针<br>WORD wBufferSize 数据大小 |
| | 返回 | |
| 举例 | | |
| | | |

- **类继承图**




### 3.3.1 运行流程






## 3.2 网络引擎 TCPSocketEngine 模块设计

### 3.2.1 网络重叠完成端口 IOCP 简介

完成端口：是一个 FIFO 队列，操作系统的 IO 子系统在 IO 操作完成后，会把相应的 IO packet 放入该队列。

-等待者线程队列：通过调用 GetQueuedCompletionStatus API，在完成端口上等待取下一个 IO packet。

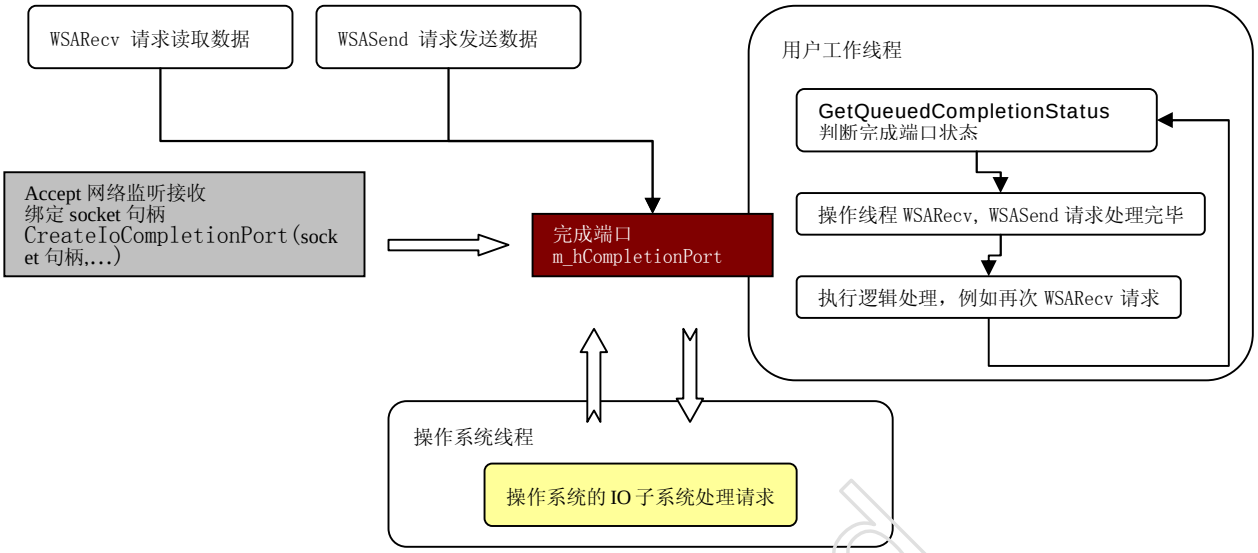
执行者线程组：已经从完成端口上获得 IO packet，在占用 CPU 进行处理。

除了以上三种类型的参与者。我们还应该注意两个关联关系，即：

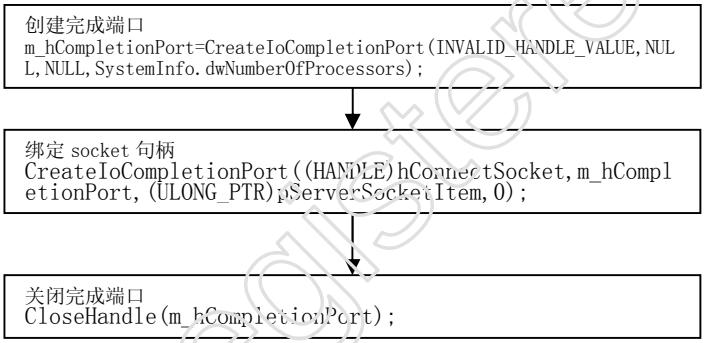
IO Handle 与完成端口相关联：任何期望使用 IOCP 的方式来处理 IO 请求的，必须将相应的 IO Handle 与该完成端口相关联。需要指出的时，这里的 IO Handle，可以是 File 的 Handle,或者是 Socket 的 Handle。

-线程与完成端口相关联：任何调用 GetQueuedCompletionStatus API 的线程，都将与该完成端口相关联。在任何给定的时候，该线程只能与一个完成端口相关联，与最后一次调用的 GetQueuedCompletionStatus 为准.

对于网络重叠完成端口体系示意图如下:




网络重叠完成端口使用示意图如下:




### 3.2.2 基本设计概念

建立数据队列服务 QueueService，对外部输入的对 socket 的发送，关闭请求等处理。同时建立工作线程 SocketAcceptThread 做应答 Socket，建立读写工作线程 ServerSocketRSThread 判断完成端口状态,等待系统线程处理读写操作完成，如果完成读操作，发送调度事件到调度引擎处理，并继续投递读操作，如果完成发送操作，判断还有发送请求就继续投递发送请求。最后建立检测工作线程，发送心跳包，检测断线 socket 连接。

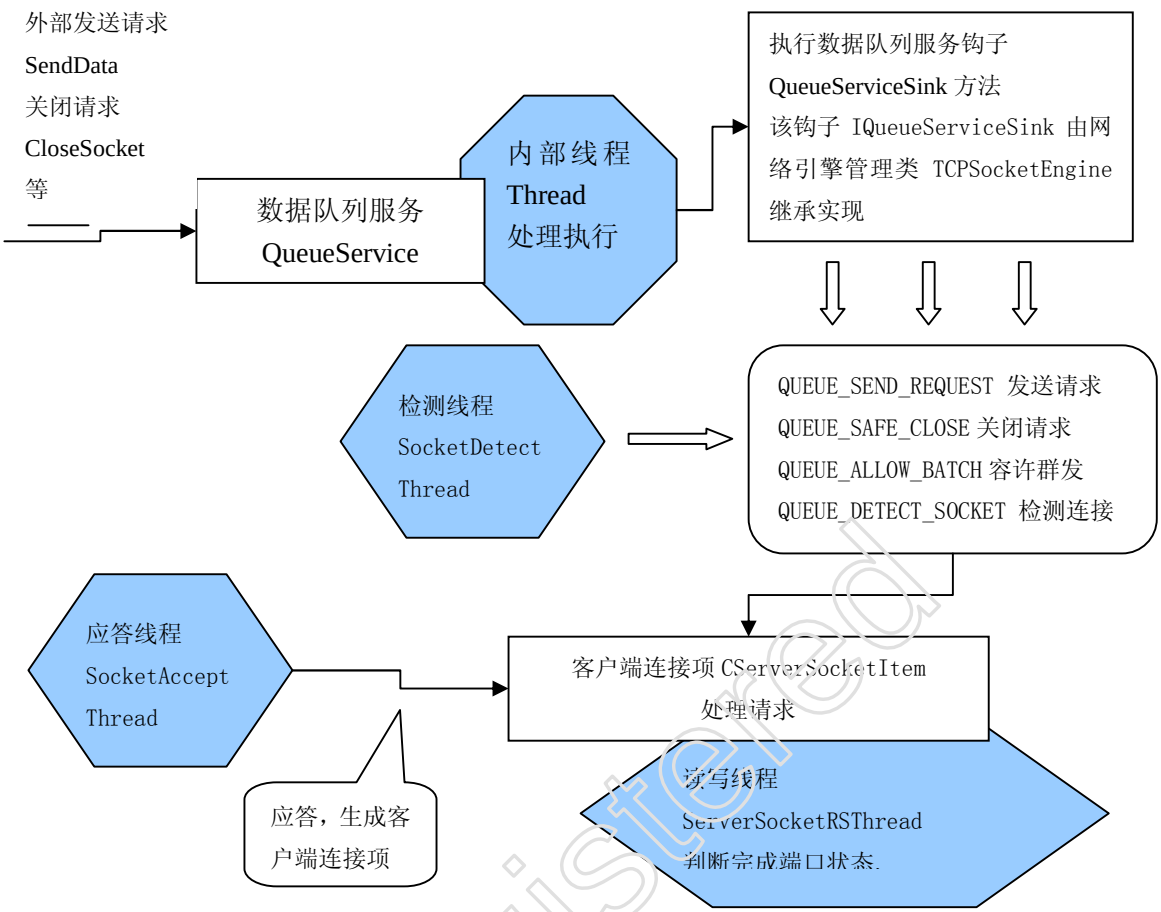
- 数据设计与管理

  先设计客户端连接项 ServerSocketItem，实现对客户端连接 Socket 的发送，接收数据，加密/解密网络包，关闭等操作

  再设计网络引擎 TCPSocketEngine，对客户端连接项的管理，同时接收外部的输入操作请求，还有对应答线程，读写线程，检测线程管理

  设计应答线程，读写线程，检测线程的功能进行封装，还有重叠数据的帮助封装。

- 工作线程运行设计



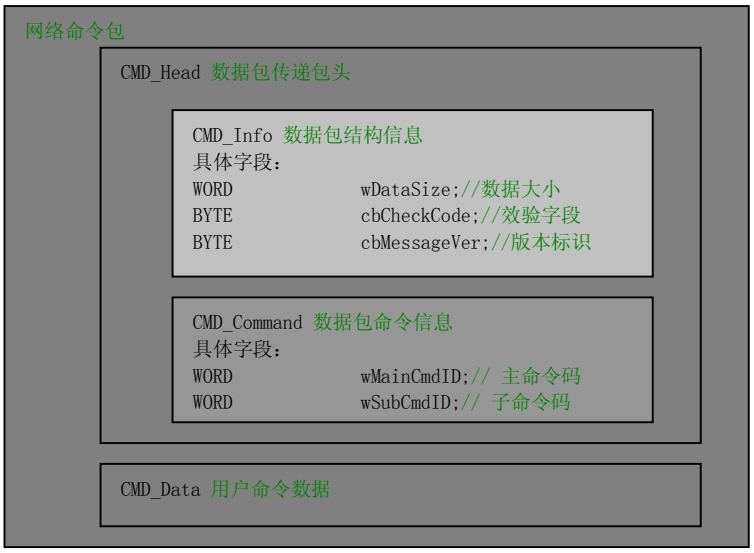

### 3.2.3 网络数据命令包结构设计

网络数据包分两个区域:包头，用户数据

包头：每个网络包都带的数据区域，装载数据包结构信息与数据包命令信息

用户数据：根据命令信息不一样，用户自定义的负载数据

具体结构如下：




### 3.2.4 网络数据加密设计

数据加密部分为CMD_Command 数据包命令信息 与 CMD_Data 用户命令数据, CMD_Info 数据包结构

信息不加密.
加密算法为根据字节影射表，将字节替换。例如数据包第6个字节是0x90,通过字节影射表查找，对应的是0x1d,则该字节替换为0x1d发送.

### 3.2.5 类与接口设计

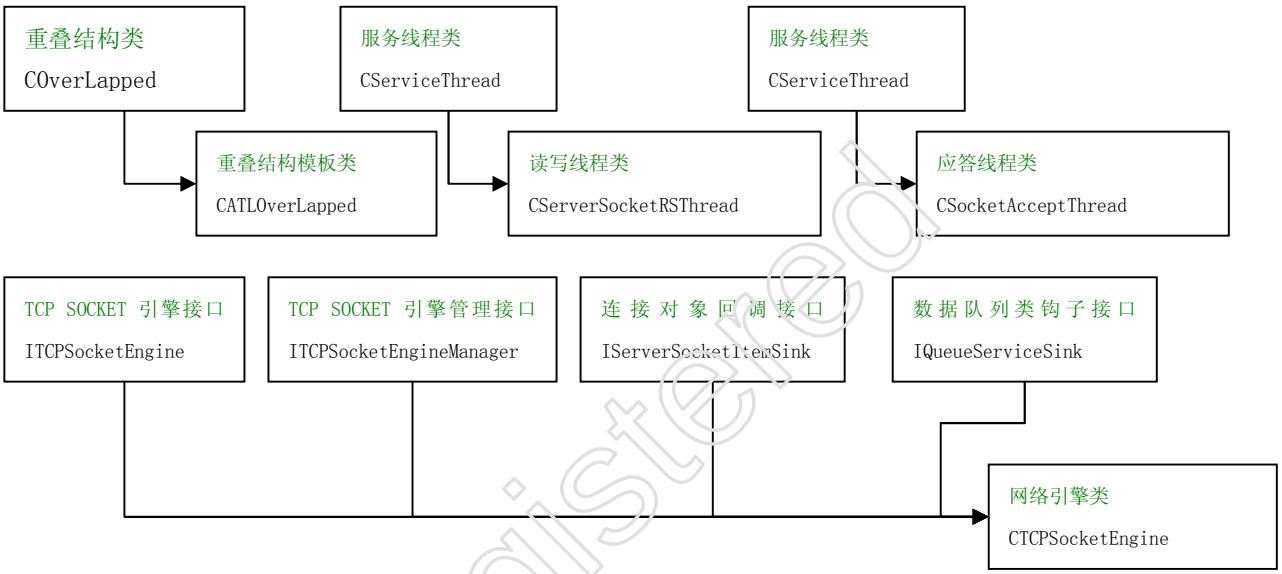
- **接口设计**

  请查阅 SocketEngine.h 头文件

- **类设计**

  请查阅 SocketEngine.h 头文件

- **类继承图**




## 3.3 数据库引擎 DataBaseEngine 模块设计

### 3.3.1 基本设计概念

数据库引擎建立数据队列服务,接受外部数据库事件数据输入,并调用外部数据库处理钩子方法处理.且提供数据库帮助类对象,实现对数据库访问,数据访问技术采用 ADO.

### 3.3.2 类与接口设计

- **接口设计**

| IADOError | | |
|---|---|---|
| 数据库错误接口 | | |
| 模块 | Database | |
| 继承 | IADOError → IUnknownEx | |
| 实现 | | |
| 数据成员 | | |
| 名称 | 说明 | |
| | | |
| 方法 | | |
| virtual LPCTSTR __cdecl GetErrorDescribe() | | |
| 说明 | 获取错误描述 | |
| | 参数 | |
| | 返回 | 描述错误的文本字符串 |

| | |
|---|---|
| 举例 | |

virtual enADOErrorType __cdecl GetErrorType()

| | | |
|---|---|---|
| 说明 | 获取错误类型 | |
| | 参数 | |
| | 返回 | 错误类型的枚举值 |
| 举例 | | |
| | | |

| IDataBase | |
|---|---|
| 数据库连接并访问接口 | |
| 模块 | Database |
| 继承 | IDataBase → IUnknownEx |
| 实现 | |

数据成员

| 名称 | 说明 |
|---|---|
| | |

方法

virtual bool __cdecl OpenConnection()

| | | |
|---|---|---|
| 说明 | 打开连接 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl CloseRecordset()

| | | |
|---|---|---|
| 说明 | 关闭记录 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl CloseConnection()

| | | |
|---|---|---|
| 说明 | 关闭连接 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl TryConnectAgain(bool bFocusConnect, CComError * pComError)

| | | | |
|---|---|---|---|
| 说明 | 重新连接 | | |
| | 参数 | bool bFocusConnect | 强制性连接标记 |
| | | CComError * pComError | 错误保存 |
| | 返回 | 操作成功返回 ture,否则 flase | |
| 举例 | | | |

virtual bool __cdecl SetConnectionInfo(LPCTSTR szIP,
LPCTSTR szPipeName,WORD wPort, LPCTSTR szData,
LPCTSTR szName, LPCTSTR szPass)

| | | |
|---|---|---|
| 说明 | 设置信息 | |
| | 参数 | LPCTSTR szIP 数据库服务器 IP 地址<br>LPCTSTR szPipeName 数据库命名实列通道<br>WORD wPort 端口<br>LPCTSTR szData 数据库名称<br>LPCTSTR szName 访问用户名<br>LPCTSTR szPass 访问密码 |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl IsConnectError()

| | | |
|---|---|---|
| 说明 | 判断是否连接错误 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl IsRecordsetOpened ()

| | | |
|---|---|---|
| 说明 | 判断是否打开记录集 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl MoveToNext ()

| | | |
|---|---|---|
| 说明 | 往下移动记录游标 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl MoveToFirst()

| | | |
|---|---|---|
| 说明 | 移到开头 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual bool __cdecl IsEndRecordset ()

| | | |
|---|---|---|
| 说明 | 是否结束 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |

virtual long __cdecl GetRecordCount ()

| | | |
|---|---|---|
| 说明 | 获取数目 | |
| | 参数 | |
| | 返回 | 操作成功返回记录数目 |
| 举例 | | |

virtual long __cdecl GetActualSize(LPCTSTR pszParamName)

| | | |
|---|---|---|
| 说明 | 获取字段大小 | |
| | 参数 | LPCTSTR pszParamName 参数名称 |
| | 返回 | 操作成功返回字段大小 |
| 举例 | | |

virtual long __cdecl BindToRecordset(CADORecordBinding * pBind)

| | | |
|---|---|---|
| 说明 | 绑定对象 | |
| | 参数 | CADORecordBinding * pBind 数据库记录绑定对象指针 |
| | 返回 | 操作成功返回 0 |
| 举例 | | |

virtual long __cdecl NextRecordset ()

| | | |
|---|---|---|
| 说明 | 下一记录集 | |
| | 参数 | |
| | 返回 | 操作成功返回 0 |
| 举例 | | |

virtual bool __cdecl GetFieldValue ()

| | | |
|---|---|---|
| 说明 | 获取记录集的数据 | |
| | 参数 | LPCTSTR lpFieldName 字段名称<br>X & bValue 字段数据值 |
| | 返回 | 操作成功返回 true |
| 举例 | | |

virtual void __cdecl SetSPName(LPCTSTR pszSpName)

| | | |
|---|---|---|
| 说明 | 获取存储过程名称 | |
| | 参数 | LPCTSTR pszSpName 存储过程名称 |
| | 返回 | |
| 举例 | | |

virtual void __cdecl AddParamter(LPCTSTR pszName, ADOCG::ParameterDirectionEnum Direction, ADOCG::DataTypeEnum Type, long lSize, _variant_t & vtValue)

| | | |
|---|---|---|
| 说明 | 添加参数 | |
| | 参数 | LPCTSTR pszName 字段名称<br>ADOCG::ParameterDirectionEnum Direction 参数输入输出方向标记<br>ADOCG::DataTypeEnum Type 参数类型<br>long lSize 参数大小<br>_variant_t & vtValue 参数值 |
| | 返回 | |
| 举例 | | |

virtual void __cdecl ClearAllParameters()

| | | |
|---|---|---|
| 说明 | 清除参数 | |
| | 参数 | |
| | 返回 | |
| 举例 | | |

| virtual void __cdecl GetParameterValue(LPCTSTR pszParamName, _variant_t & vtValue) | | |
|---|---|---|
| 说明 | 获取参数值 | |
| | 参数 | LPCTSTR pszParamName 参数名称<br>_variant_t & vtValue 参数返回值 |
| | 返回 | |
| 举例 | | |

| virtual long __cdecl GetReturnValue() | | |
|---|---|---|
| 说明 | 获取数据库返回值 | |
| | 参数 | |
| | 返回 | |
| 举例 | | |

| virtual bool __cdecl Execute(LPCTSTR pszCommand) | | |
|---|---|---|
| 说明 | 执行语句 | |
| | 参数 | LPCTSTR pszCommand SQL 语句 |
| | 返回 | 操作成功返回 ture,否则为 false |
| 举例 | | |

| virtual bool __cdecl ExecuteCommand(bool bRecordset) | | |
|---|---|---|
| 说明 | 执行命令 | |
| | 参数 | bool bRecordset 是否返回记录集 |
| | 返回 | 操作成功返回 ture,否则为 false |
| 举例 | | |
| | | |

| IDataBaseSink | |
|---|---|
| 数据库钩子接口 | |
| 模块 | Database |
| 继承 | IDataBaseSink → IUnknownEx |
| 实现 | |

| 数据成员 | |
|---|---|
| 名称 | 说明 |
| | |

| 方法 | | |
|---|---|---|
| virtual bool __cdecl StartService(IUnknownEx * pIUnknownEx) | | |
| 说明 | 数据库模块启动 | |
| | 参数 | IUnknownEx * pIUnknownEx 服务框架接口 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

| virtual bool __cdecl StopService(IUnknownEx * pIUnknownEx) | | |
|---|---|---|
| 说明 | 数据库模块关闭 | |
| | 参数 | IUnknownEx * pIUnknownEx 服务框架接口 |

| | | |
|---|---|---|
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl OnDataBaseRequest(BYTE cbThreadIndex,const NTY_DataBaseEvent & DataBaseEvent, void * pDataBuffer, WORD wDataSize)

| | | |
|---|---|---|
| 说明 | 数据事件操作处理 | |
| | 参数 | BYTE cbThreadIndex 线程索引<br>const NTY_DataBaseEvent & DataBaseEvent 数据库事件<br>void * pDataBuffer 数据指针<br>WORD wDataSize 数据大小 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| | | |

| IDataBaseEngine | |
|---|---|
| 数据库引擎接口 | |
| 模块 | Database |
| 继承 | IDataBaseEngine → IUnknownEx |
| 实现 | |

数据成员

| 名称 | 说明 |
|---|---|
| | |

方法

virtual bool __cdecl StartService(BYTE cbThreadCount)

| | | |
|---|---|---|
| 说明 | 启动服务 | |
| | 参数 | BYTE cbThreadCount 数据库线程数量 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl StopService()

| | | |
|---|---|---|
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetDataBaseSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置数据事件操作处理钩子 | |
| | 参数 | IUnknownEx * pIUnknownEx 钩子指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual void * __cdecl GetQueueService(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取接口 | |
| | 参数 | onst IID & Guid 接口标记<br>DWORD dwQueryVer 查询版本 |

| | | |
|---|---|---|
| | 返回 | 操作成功返回内存指针,否则 Null |
| 举例 | | |
| | | |

- **类设计**

| CADOError | | |
|---|---|---|
| 数据库错误类 | | |
| 模块 | Database | |
| 继承 | CADOError → IADOError → IUnknownEx | |
| 实现 | IADOError | |
| 数据成员 | | |
| 名称 | | 说明 |
| enADOErrorType m_enErrorType | | 错误代号 |
| CString m_strErrorDescribe; | | 错误信息 |
| 方法 | | |
| virtual LPCTSTR __cdecl GetErrorDescribe() | | |
| 说明 | 获取错误描述 | |
| | 参数 | |
| | 返回 | 描述错误的文本字符串 |
| 举例 | | |
| virtual enADOErrorType __cdecl GetErrorType() | | |
| 说明 | 获取错误类型 | |
| | 参数 | |
| | 返回 | 错误类型的枚举值 |
| 举例 | | |
| void SetErrorInfo(enADOErrorType enErrorType, LPCTSTR pszDescribe) | | |
| 说明 | 设置错误 | |
| | 参数 | enADOErrorType enErrorType 错误类型的枚举值<br>LPCTSTR pszDescribe 描述错误的文本字符串 |
| | 返回 | |
| 举例 | | |
| | | |

| CDataBase | | |
|---|---|---|
| 数据库连接并访问类 | | |
| 模块 | Database | |
| 继承 | CDataBase → IDataBase → IUnknownEx | |
| 实现 | IDataBase | |
| 数据成员 | | |
| 名称 | | 说明 |

| | |
|---|---|
| CADOError m_ADOError; | 错误对象 |
| CString m_strConnect; | 连接字符串 |
| CString m_strErrorDescribe; | 错误信息 |
| DWORD m_dwConnectCount; | 重试次数 |
| DWORD m_dwConnectErrorTime; | 错误时间 |
| const DWORD m_dwResumeConnectCount; | 恢复次数 |
| const DWORD m_dwResumeConnectTime; | 恢复时间 |
| _CommandPtr m_DBCommand; | 命令对象 |
| _RecordsetPtr m_DBRecordset; | 记录集对象 |
| _ConnectionPtr m_DBConnection; | 数据库连接对象 |

方法

virtual bool __cdecl OpenConnection()

| | | |
|---|---|---|
| 说明 | 打开连接 | |
| 参数 | | |
| 返回 | 操作成功返回 ture,否则 flase | |
| 举例 | | |

virtual bool __cdecl CloseRecordset()

| | | |
|---|---|---|
| 说明 | 关闭记录 | |
| 参数 | | |
| 返回 | 操作成功返回 ture,否则 flase | |
| 举例 | | |

virtual bool __cdecl CloseConnection()

| | | |
|---|---|---|
| 说明 | 关闭连接 | |
| 参数 | | |
| 返回 | 操作成功返回 ture,否则 flase | |
| 举例 | | |

virtual bool __cdecl TryConnectAgain(bool bFocusConnect, CComError * pComError)

| | | |
|---|---|---|
| 说明 | 重新连接 | |
| 参数 | bool bFocusConnect | 强制性连接标记 |
| | CComError * pComError | 错误保存 |
| 返回 | 操作成功返回 ture,否则 flase | |
| 举例 | | |

virtual bool __cdecl SetConnectionInfo(LPCTSTR szIP, LPCTSTR szPipeName,WORD wPort, LPCTSTR szData, LPCTSTR szName, LPCTSTR szPass)

| | | |
|---|---|---|
| 说明 | 设置信息 | |
| 参数 | LPCTSTR szIP | 数据库服务器 IP 地址 |
| | LPCTSTR szPipeName | 数据库命名实列通道 |
| | WORD wPort | 端口 |
| | LPCTSTR szData | 数据库名称 |
| | LPCTSTR szName | 访问用户名 |

| | | |
|---|---|---|
| | | LPCTSTR szPass 访问密码 |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |
| virtual bool __cdecl IsConnectError() | | |
| 说明 | 判断是否连接错误 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |
| virtual bool __cdecl IsRecordsetOpened () | | |
| 说明 | 判断是否打开记录集 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |
| virtual bool __cdecl MoveToNext () | | |
| 说明 | 往下移动记录游标 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |
| virtual bool __cdecl MoveToFirst() | | |
| 说明 | 移到开头 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |
| virtual bool __cdecl IsEndRecordset () | | |
| 说明 | 是否结束 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 flase |
| 举例 | | |
| virtual long __cdecl GetRecordCount () | | |
| 说明 | 获取数目 | |
| | 参数 | |
| | 返回 | 操作成功返回记录数目 |
| 举例 | | |
| virtual long __cdecl GetActualSize(LPCTSTR pszParamName) | | |
| 说明 | 获取字段大小 | |
| | 参数 | LPCTSTR pszParamName 参数名称 |
| | 返回 | 操作成功返回字段大小 |
| 举例 | | |
| virtual long __cdecl BindToRecordset(CADORecordBinding * pBind) | | |
| 说明 | 绑定对象 | |

| | | |
|---|---|---|
| | 参数 | CADORecordBinding * pBind　　数据库记录绑定对象指针 |
| | 返回 | 操作成功返回 0 |
| 举例 | | |

virtual long __cdecl NextRecordset ()

| | | |
|---|---|---|
| 说明 | 下一记录集 | |
| | 参数 | |
| | 返回 | 操作成功返回 0 |
| 举例 | | |

virtual bool __cdecl GetFieldValue ()

| | | |
|---|---|---|
| 说明 | 获取记录集的数据 | |
| | 参数 | LPCTSTR lpFieldName　　字段名称<br>X & bValue　　字段数据值 |
| | 返回 | 操作成功返回 true |
| 举例 | | |

virtual void __cdecl SetSPName(LPCTSTR pszSpName)

| | | |
|---|---|---|
| 说明 | 获取存储过程名称 | |
| | 参数 | LPCTSTR pszSpName　　存储过程名称 |
| | 返回 | |
| 举例 | | |

virtual void __cdecl AddParamter(LPCTSTR pszName, ADOCG::ParameterDirectionEnum Direction, ADOCG::DataTypeEnum Type, long lSize, _variant_t & vtValue)

| | | |
|---|---|---|
| 说明 | 添加参数 | |
| | 参数 | LPCTSTR pszName　　字段名称<br>ADOCG::ParameterDirectionEnum Direction 参数输入输出方向标记<br>ADOCG::DataTypeEnum Type　参数类型<br>long lSize　　参数大小<br>_variant_t & vtValue　　参数值 |
| | 返回 | |
| 举例 | | |

virtual void __cdecl ClearAllParameters()

| | | |
|---|---|---|
| 说明 | 清除参数 | |
| | 参数 | |
| | 返回 | |
| 举例 | | |

virtual void __cdecl GetParameterValue(LPCTSTR pszParamName, _variant_t & vtValue)

| | | |
|---|---|---|
| 说明 | 获取参数值 | |
| | 参数 | LPCTSTR pszParamName　　参数名称<br>_variant_t & vtValue　　参数返回值 |
| | 返回 | |
| 举例 | | |

virtual long __cdecl GetReturnValue()

| | | |
|---|---|---|
| 说明 | 获取数据库返回值 | |
| | 参数 | |
| | 返回 | |
| 举例 | | |

virtual bool __cdecl Execute(LPCTSTR pszCommand)

| | | |
|---|---|---|
| 说明 | 执行语句 | |
| | 参数 | LPCTSTR pszCommand　　SQL 语句 |
| | 返回 | 操作成功返回 ture,否则为 false |
| 举例 | | |

virtual bool __cdecl ExecuteCommand(bool bRecordset)

| | | |
|---|---|---|
| 说明 | 执行命令 | |
| | 参数 | bool bRecordset　　是否返回记录集 |
| | 返回 | 操作成功返回 ture,否则为 false |
| 举例 | | |

LPCTSTR GetComErrorDescribe(CComError & ComError)

| | | |
|---|---|---|
| 说明 | 获取错误描述 | |
| | 参数 | CComError & ComError　　错误对象 |
| | 返回 | 操作成功返回文本 |
| 举例 | | |

void SetErrorInfo(enADOErrorType enErrorType, LPCTSTR pszDescribe)

| | | |
|---|---|---|
| 说明 | 设置错误 | |
| | 参数 | enADOErrorType enErrorType　　错误类型<br>LPCTSTR pszDescribe　　错误描述 |
| | 返回 | |
| 举例 | | |
| | | |

CDataBaseEngine

数据库引擎类

| | |
|---|---|
| 模块 | Database |
| 继承 | CDataBaseEngine → IDataBaseEngine → IUnknownEx |
| 实现 | IDataBaseEngine |

数据成员

| 名称 | 说明 |
|---|---|
| bool　　m_bService; | 运行标志 |
| CQueueService m_RequestQueueService; | 队列对象 |
| IDataBaseSink *m_pIDataBaseSink; | 通知钩子 |

方法

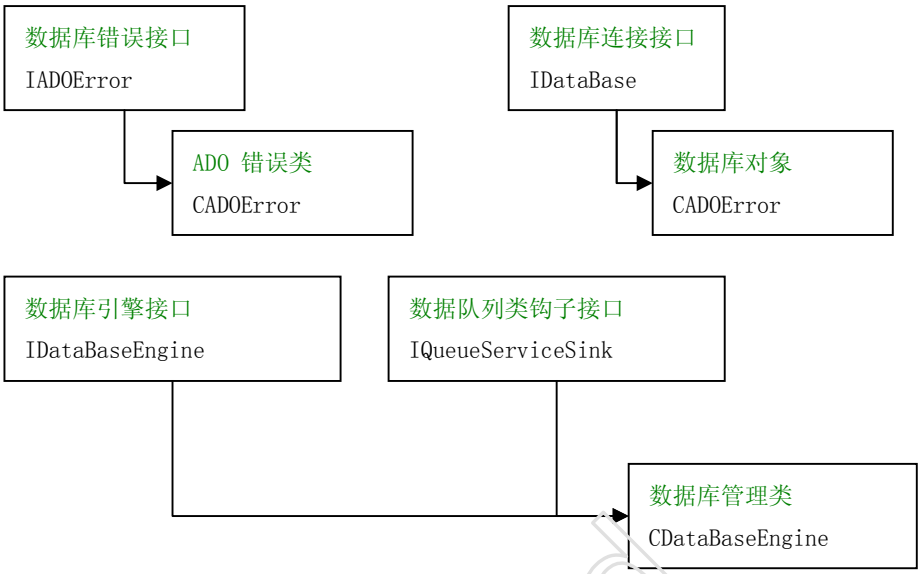
virtual bool __cdecl StartService(BYTE cbThreadCount)

| | | |
|---|---|---|
| 说明 | 启动服务 | |
| | 参数 | BYTE cbThreadCount　　数据库线程数量 |

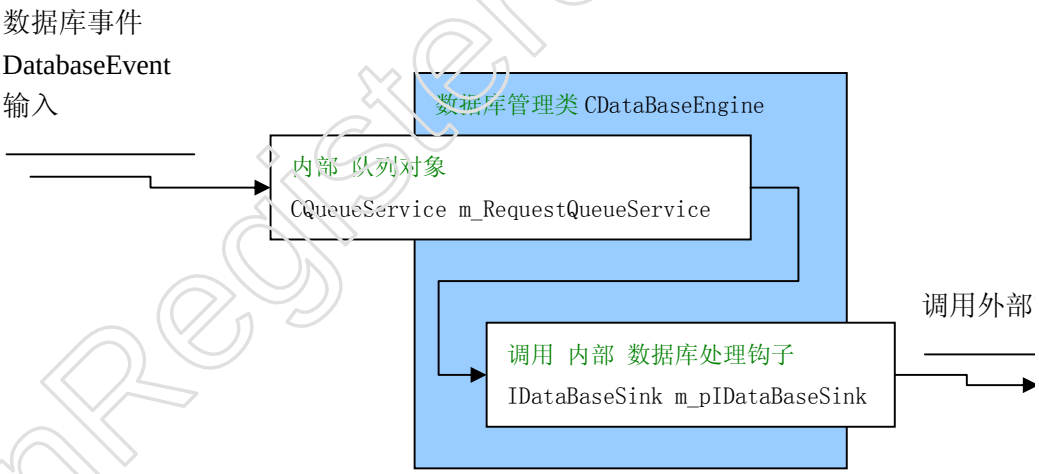
| | | |
|---|---|---|
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl StopService() | | |
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl SetDataBaseSink(IUnknownEx * pIUnknownEx) | | |
| 说明 | 设置数据事件操作处理钩子 | |
| | 参数 | IUnknownEx * pIUnknownEx 钩子指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual void * __cdecl GetQueueService(const IID & Guid, DWORD dwQueryVer) | | |
| 说明 | 获取接口 | |
| | 参数 | onst IID & Guid 接口标记<br>DWORD dwQueryVer 查询版本 |
| | 返回 | 操作成功返回内存指针,否则 Null |
| 举例 | | |
| virtual void __cdecl OnQueueServiceSink(BYTE cbThreadIndex,WORD wIdentifier, void * pBuffer, WORD wDataSize, DWORD dwInsertTime) | | |
| 说明 | 实现队列钩子接口的函数 | |
| | 参数 | BYTE cbThreadIndex 线程索引<br>WORD wIdentifier 事件数据标记<br>void * pBuffer 数据指针<br>WORD wDataSize 数据大小<br>DWORD dwInsertTime 加入时间 |
| | 返回 | |
| 举例 | | |
| | | |



- **类继承图**




### 3.3.3 运行流程






## 3.4 定时器引擎 TimerEngine 模块设计

### 3.4.1 基本设计概念

内部启动工作线程 TimerThread，工作线程周期检测定时请求子项 tagTimerItem，每个定时请求项记录当前时间计数 dwTimeLeave,处理一次，将时间计数减去时间间隔 dwTimerSpace，如果为 0 时,发生定时器事件 TimerEvent,通知到调度引擎,并将时间计数复位为定时时间再次倒计时.

### 3.4.2 类与接口设计

- **接口设计**

| ITimerEngine | |
|---|---|
| 定时器引擎接口 | |
| 模块 | TimerEngine |
| 继承 | ITimerEngine → IUnknownEx |
| 实现 | |

| 数据成员 | |
|---|---|
| 名称 | 说明 |
| | |

| 方法 | | | |
|---|---|---|---|
| virtual bool __cdecl SetTimer(WORD wTimerID, DWORD dwElapse, DWORD dwRepeat, WPARAM wParam) | | | |
| 说明 | 设置定时器 | | |
| | 参数 | WORD wTimerID | 时间标记 |
| | | DWORD dwElapse | 时间间隔 |
| | | DWORD dwRepeat | 重复次数 |
| | | WPARAM wParam | 绑定参数 |
| | 返回 | 操作成功返回 ture,否则 false | |
| 举例 | | | |
| virtual bool __cdecl KillTimer(WORD wTimerID) | | | |
| 说明 | 删除定时器 | | |
| | 参数 | WORD wTimerID | 时间标记 |
| | 返回 | 操作成功返回 ture,否则 false | |
| 举例 | | | |
| virtual bool __cdecl KillAllTimer() | | | |
| 说明 | 删除全部定时器 | | |
| | 参数 | | |
| | 返回 | 操作成功返回 ture,否则 false | |
| 举例 | | | |
| | | | |

| ITimerEngineManager | |
|---|---|
| 定时器引擎管理接口 | |
| 模块 | TimerEngine |
| 继承 | ITimerEngineManager → IUnknownEx |
| 实现 | |

| 数据成员 | |
|---|---|
| 名称 | 说明 |
| | |

| 方法 | | | |
|---|---|---|---|
| virtual bool __cdecl StartService() | | | |
| 说明 | 启动服务 | | |
| | 参数 | | |
| | 返回 | 操作成功返回 ture,否则 false | |
| 举例 | | | |
| virtual bool __cdecl StopService() | | | |
| 说明 | 停止服务 | | |

| | | |
|---|---|---|
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetTimerEngineSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置时间引擎钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 调度钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| | | |

- **类设计**

| CTimerEngine | |
|---|---|
| 定时器引擎 | |
| 模块 | TimerEngine |
| 继承 | CTimerEngine → ITimerEngine, ITimerEngineManager → IUnknownEx |
| 实现 | ITimerEngine ,ITimerEngineManager |

数据成员

| 名称 | 说明 |
|---|---|
| DWORD m_dwTimerSpace; | 时间间隔 |
| bool m_bService; | 运行标志 |
| DWORD m_dwTimePass; | 经过时间 |
| DWORD m_dwTimeLeave; | 倒计时间 |
| CTimerItemPtr m_TimerItemFree; | 空闲数组 |
| CTimerItemPtr m_TimerItemActive; | 活动数组 |
| CThreadLock m_ThreadLock; | 线程锁 |
| CTimerThread m_TimerThread; | 定时器线程 |
| CQueueServiceEvent m_AttemperEvent; | 通知组件 |

方法

virtual bool __cdecl SetTimer(WORD wTimerID, DWORD dwElapse, DWORD dwRepeat, WPARAM wParam)

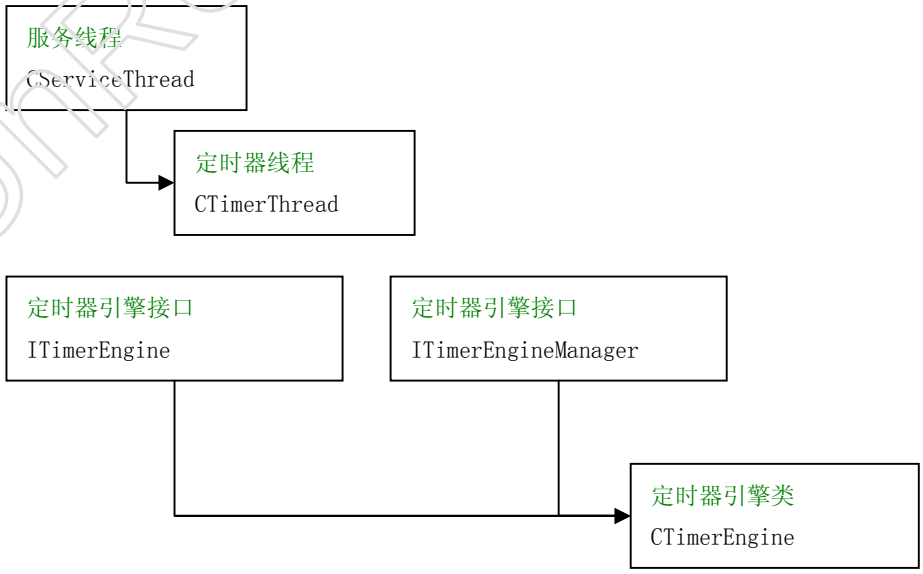
| | | | |
|---|---|---|---|
| 说明 | 设置定时器 | | |
| | 参数 | WORD wTimerID | 时间标记 |
| | | DWORD dwElapse | 时间间隔 |
| | | DWORD dwRepeat | 重复次数 |
| | | WPARAM wParam | 绑定参数 |
| | 返回 | 操作成功返回 ture,否则 false | |
| 举例 | | | |

virtual bool __cdecl KillTimer(WORD wTimerID)

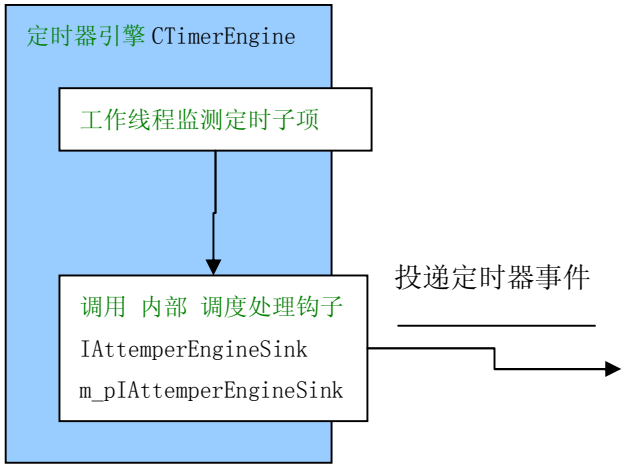
| | | | |
|---|---|---|---|
| 说明 | 删除定时器 | | |
| | 参数 | WORD wTimerID | 时间标记 |
| | 返回 | 操作成功返回 ture,否则 false | |
| 举例 | | | |

| virtual bool __cdecl KillAllTimer() | | |
|---|---|---|
| 说明 | 删除全部定时器 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl StartService() | | |
| 说明 | 启动服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl StopService() | | |
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl SetTimerEngineSink(IUnknownEx * pIUnknownEx) | | |
| 说明 | 设置时间引擎钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 调度钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| | | |

- **类继承图**




### 3.4.3 运行流程




### 3.5 调度引擎 AttemperEngine 模块设计

#### 3.5.1 基本设计概念

内部管理数据队列,对外提供接口,接收外部输入网络事件,数据库事件,定时器事件等数据,对这些事件做初步处理,并调度分发传递到外部调度引擎钩子,让外部实现逻辑处理

#### 3.5.2 类与接口设计

- **接口设计**

| IAttemperEngine | | |
|---|---|---|
| 调度引擎钩子接口 | | |
| 模块 | AttemperEngine | |
| 继承 | IAttemperEngine → IUnknownEx | |
| 实现 | | |
| 数据成员 | | |
| 名称 | 说明 | |
| | | |
| 方法 | | |
| virtual bool __cdecl StartService(BYTE cbThreadCount) | | |
| 说明 | 启动服务 | |
| | 参数 | BYTE cbThreadCount 线程数量 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl StopService() | | |
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl SetSocketEngine(IUnknownEx * pIUnknownEx) | | |
| 说明 | 设置网络引擎接口 | |

| | | |
|---|---|---|
| | 参数 | IUnknownEx * pIUnknownEx 网络引擎接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetAttemperEngineSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置调度引擎钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 调度钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual void * __cdecl GetQueueService(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取数据队列接口 | |
| | 参数 | const IID & Guid 接口标记<br>DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回数据队列接口 |
| 举例 | | |
| | | |

| IAttemperEngineSink | |
|---|---|
| 调度钩子接口 | |
| 模块 | AttemperEngine |
| 继承 | IAttemperEngineSink --> IUnknownEx |
| 实现 | |

数据成员

| 名称 | 说明 |
|---|---|
| | |

方法

virtual bool __cdecl StartService(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 调度模块启动 | |
| | 参数 | IUnknownEx * pIUnknownEx 服务框架接口 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl StopService(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 调度模块关闭 | |
| | 参数 | IUnknownEx * pIUnknownEx 服务框架接口 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl OnAttemperEvent (BYTE cbThreadIndex,WORD wIdentifier, void * pBuffer, WORD wDataSize, DWORD dwInsertTime)

| | | |
|---|---|---|
| 说明 | 调度事件操作处理 | |
| | 参数 | BYTE cbThreadIndex 线程索引<br>WORD wIdentifier 调度事件标记<br>void * pDataBuffer 数据指针 |

| | | |
|---|---|---|
| | | WORD wDataSize 数据大小<br>DWORD dwInsertTime 插入时间 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl OnEventTimer(BYTE cbThreadIndex,WORD wTimerID, WPARAM wBindParam)

| | | |
|---|---|---|
| 说明 | 定时器事件 | |
| | 参数 | BYTE cbThreadIndex 线程索引<br>WORD wTimerID 定时器标记<br>WPARAM wBindParam 绑定参数 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl OnEventDataBase(BYTE cbThreadIndex,void * pDataBuffer, WORD wDataSize, NTY_DataBaseEvent * pDataBaseEvent)

| | | |
|---|---|---|
| 说明 | 数据库事件 | |
| | 参数 | BYTE cbThreadIndex 线程索引<br>void * pDataBuffer 数据指针<br>WORD wDataSize 数据大小<br>NTY_DataBaseEvent * pDataBaseEvent 数据库事件 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl OnEventSocketAccept(BYTE cbThreadIndex,NTY_SocketAcceptEvent * pSocketAcceptEvent)

| | | |
|---|---|---|
| 说明 | 网络应答事件 | |
| | 参数 | BYTE cbThreadIndex 线程索引<br>NTY_SocketAcceptEvent * pSocketAcceptEvent 网络应答事件 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl OnEventSocketRead(BYTE cbThreadIndex,CMD_Command Command, void * pDataBuffer, WORD wDataSize, NTY_SocketReadEvent * pSocketReadEvent)

| | | |
|---|---|---|
| 说明 | 网络应答事件 | |
| | 参数 | BYTE cbThreadIndex 线程索引<br>CMD_Command Command 网络命令<br>void * pDataBuffer 数据指针<br>WORD wDataSize 数据大小<br>NTY_SocketReadEvent * pSocketReadEvent 网络读取事件 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl OnEventSocketClose(BYTE cbThreadIndex,NTY_SocketCloseEvent * pSocketCloseEvent)

| | |
|---|---|
| 说明 | 网络应答事件 |

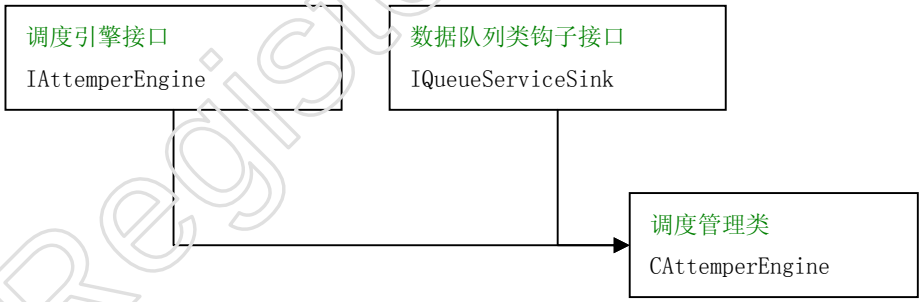
| | | |
|---|---|---|
| | 参数 | BYTE cbThreadIndex 线程索引<br>NTY_SocketCloseEvent * pSocketCloseEvent 网络读取事件 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| | | |

- **类设计**

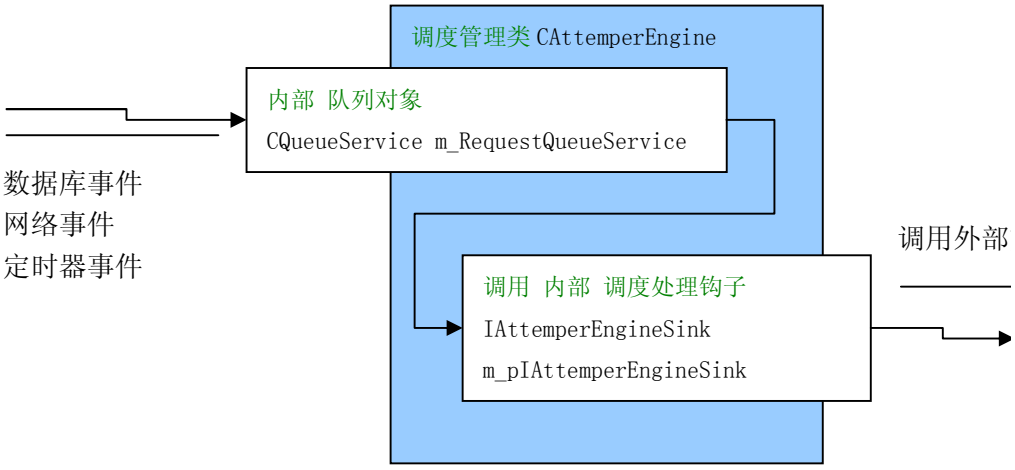
| CAttemperEngine | | |
|---|---|---|
| 调度引擎类 | | |
| 模块 | AttemperEngine | |
| 继承 | CAttemperEngine → IAttemperEngine → IUnknownEx | |
| 实现 | IAttemperEngine | |
| 数据成员 | | |
| 名称 | | 说明 |
| bool m_bService; | | 运行标志 |
| CQueueService m_RequestQueueService; | | 队列对象 |
| ITCPSocketEngine* m_pITCPSocketEngine | | 网络引擎 |
| IAttemperEngineSink* m_pIAttemperEngineSink; | | 挂接接口 |
| 方法 | | |
| virtual bool __cdecl StartService(BYTE cbThreadCount) | | |
| 说明 | 启动服务 | |
| | 参数 | BYTE cbThreadCount 线程数量 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl StopService() | | |
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl SetSocketEngine(IUnknownEx * pIUnknownEx) | | |
| 说明 | 设置网络引擎接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 网络引擎接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl SetAttemperEngineSink(IUnknownEx * pIUnknownEx) | | |
| 说明 | 设置调度引擎钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 调度钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual void * __cdecl GetQueueService(const IID & Guid, DWORD dwQueryVer) | | |
| 说明 | 获取数据队列接口 | |

| | | | |
|---|---|---|---|
| | 参数 | const IID & Guid | 接口标记 |
| | | DWORD dwQueryVer | 接口版本 |
| | 返回 | 操作成功返回数据队列接口 | |
| 举例 | | | |
| virtual void __cdecl OnQueueServiceSink(BYTE cbThreadIndex,WORD wIdentifier, void * pBuffer, WORD wDataSize, DWORD dwInsertTime) | | | |
| 说明 | 实现队列钩子接口的函数 | | |
| | 参数 | BYTE cbThreadIndex | 线程索引 |
| | | WORD wIdentifier | 事件数据标记 |
| | | void * pBuffer | 数据指针 |
| | | WORD wDataSize | 数据大小 |
| | | DWORD dwInsertTime | 加入时间 |
| | 返回 | | |
| 举例 | | | |
| | | | |

- **类继承图**




### 3.5.3 运行流程




## 3.6 异步引擎 AsynchronismEngine 模块设计

### 3.6.1 基本设计概念

接收请求事件,内部启动线程,线程运行异步获取请求事件,再调用钩子处理事件.异步方式采用 window 窗口消息实现.

### 3.6.2 类与接口设计

- 接口设计

| IAsynchronismEngine | | |
|---|---|---|
| 异步引擎接口 | | |
| 模块 | AsynchronismEngine | |
| 继承 | IAsynchronismEngine → IUnknownEx | |
| 实现 | | |
| 数据成员 | | |
| 名称 | 说明 | |
| | | |
| 方法 | | |
| virtual bool __cdecl StartService() | | |
| 说明 | 启动服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl StopService() | | |
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl InsertRequest(WORD wRequestID, void * const pBuffer, WORD wDataSize, IUnknownEx * pIUnknownEx) | | |
| 说明 | 数据事件操作处理 | |
| | 参数 | WORD wRequestID 请求标记<br>void * const pBuffer 数据库指针<br>WORD wDataSize 数据大小<br>IUnknownEx * pIUnknownEx 引擎处理钩子 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl RegisterAsynchronismEngineSink(IUnknownEx * pIUnknownEx) | | |
| 说明 | 注册调度钩子 | |
| | 参数 | IUnknownEx * pIUnknownEx 引擎处理钩子 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl UnRegisterAsynchronismEngineSink(IUnknownEx * pIUnknownEx) | | |
| 说明 | 取消注册 | |
| | 参数 | IUnknownEx * pIUnknownEx 引擎处理钩子 |

| | | |
|---|---|---|
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| | | |

| IAsynchronismEngineSink | | |
|---|---|---|
| 异步引擎钩子接口 | | |
| 模块 | AsynchronismEngine | |
| 继承 | IAsynchronismEngineSink → IUnknownEx | |
| 实现 | | |
| 数据成员 | | |
| 名称 | 说明 | |
| | | |
| 方法 | | |
| virtual bool __cdecl OnAsynchronismEngineStart() | | |
| 说明 | 启动异步引擎事件调用该方法 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl OnAsynchronismEngineStop() | | |
| 说明 | 停止异步引擎事件调用该方法 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl OnAsynchronismRequest(WORD wRequestID, void * pBuffer, WORD wDataSize) | | |
| 说明 | 异步请求 | |
| | 参数 | WORD wRequestID 请求标记<br>void * const pBuffer 数据库指针<br>WORD wDataSize 数据大小 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| | | |

- 类设计

| CAsynchronismEngine | |
|---|---|
| 异步引擎类 | |
| 模块 | AsynchronismEngine |
| 继承 | CAsynchronismEngine → IAsynchronismEngine → IUnknownEx |
| 实现 | IAsynchronismEngine |
| 数据成员 | |

| 名称 | 说明 |
| --- | --- |
| bool m_bService; | 服务标志 |
| BYTE m_cbBuffer[MAX_QUEUE_PACKET]; | 数据缓冲 |
| CAsynchronismEngineSinkArray m_AsynchronismEngineSinkArray; | 异步钩子 |
| CControlWnd m_ControlWnd; | 控制窗口 |
| CThreadLock m_ThreadLock; | 线程同步 |
| CDataStorage m_DataStorage; | 数据存储 |
| CMessageThread m_MessageThread; | 线程组件 |

**方法**

**virtual bool __cdecl StartService()**

| | | |
| --- | --- | --- |
| 说明 | 启动服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

**virtual bool __cdecl StopService()**

| | | |
| --- | --- | --- |
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

**virtual bool __cdecl InsertRequest(WORD wRequestID, void * const pBuffer, WORD wDataSize, IUnknownEx * pIUnknownEx)**

| | | |
| --- | --- | --- |
| 说明 | 数据事件操作处理 | |
| | 参数 | WORD wRequestID 请求标记<br>void * const pBuffer 数据库指针<br>WORD wDataSize 数据大小<br>IUnknownEx * pIUnknownEx 引擎处理钩子 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

**virtual bool __cdecl RegisterAsynchronismEngineSink(IUnknownEx * pIUnknownEx)**

| | | |
| --- | --- | --- |
| 说明 | 注册调度钩子 | |
| | 参数 | IUnknownEx * pIUnknownEx 引擎处理钩子 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

**virtual bool __cdecl UnRegisterAsynchronismEngineSink(IUnknownEx * pIUnknownEx)**

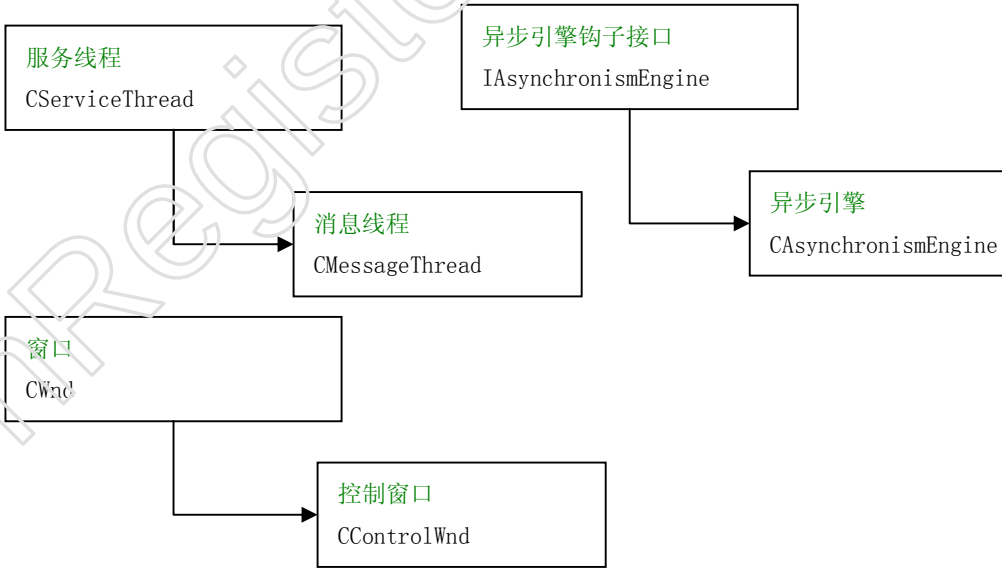
| | | |
| --- | --- | --- |
| 说明 | 取消注册 | |
| | 参数 | IUnknownEx * pIUnknownEx 引擎处理钩子 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

**bool OnMessageThreadStart()**

| | | |
| --- | --- | --- |
| 说明 | 线程启动 | |
| | 参数 | |

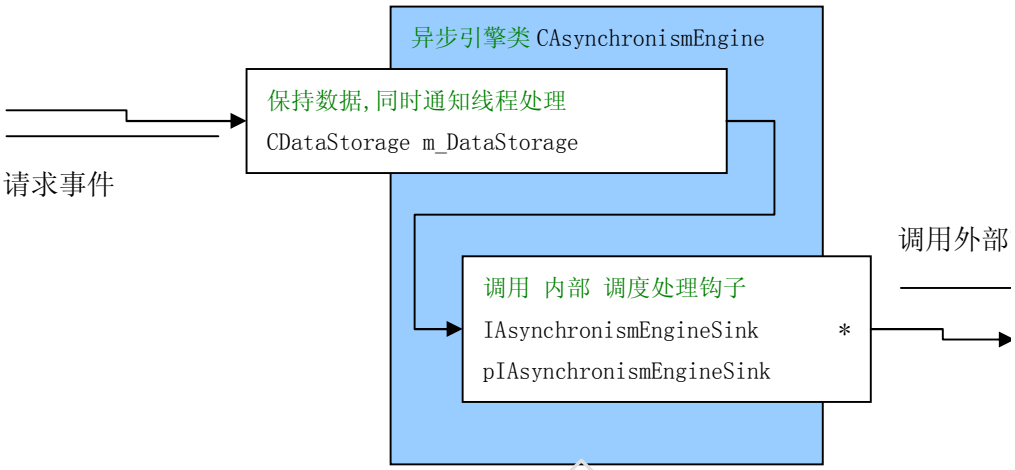
| | | |
|---|---|---|
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| bool OnMessageThreadStop () | | |
| 说明 | 线程停止 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| void OnAsynchronismRequest(WORD wRequestID, IAsynchronismEngineSink * pIAsynchronismEngineSink) | | |
| 说明 | 异步请求 | |
| | 参数 | WORD wRequestID 请求标记<br>IAsynchronismEngineSink＊pIAsynchronismEngineSink 异步引擎钩子接口 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| | | |

- 类继承图






### 3.6.3 运行流程




## 3.7 服务引擎 ServiceEngine 模块设计

### 3.7.1 基本设计概念

管理定时器引擎,数据库引擎,调度引擎,网络引擎,异步引擎,并且对外提供查询引擎接口

### 3.7.2 类与接口设计

- **接口设计**

| IServiceEngine | | |
|---|---|---|
| 服务引擎接口 | | |
| 模块 | ServiceEngine | |
| 继承 | IServiceEngine → IUnknownEx | |
| 实现 | | |
| 数据成员 | | |
| 名称 | 说明 | |
| | | |
| 方法 | | |
| virtual bool __cdecl StartService(BYTE cbDBThreadCount = 12, BYTE cbAttemperThreadCount = 1) | | |
| 说明 | 启动服务 | |
| | 参数 | BYTE cbDBThreadCount 数据库线程数量<br>BYTE cbAttemperThreadCount 调度引擎线程数量 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl StopService() | | |
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl SetEventService(IUnknownEx * pIUnknownEx) | | |

| | | |
|---|---|---|
| 说明 | 设置事件服务接口 | |
| | 参数 | IUnknownEx * pIUnknownEx　　事件服务接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetDataBaseSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置数据库钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx　　数据库钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetAttemperEngineSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置调度钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx　　调度钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl RegisterAsynchronismEngineSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 注册异步处理钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx　　异步调用钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl InitServiceEngine(WORD wListenPort, WORD wMaxSocketItem)

| | | |
|---|---|---|
| 说明 | 设置网络引擎接口 | |
| | 参数 | WORD wListenPort　　监听端口<br>WORD wMaxSocketItem　　最大连接数目 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual void * __cdecl GetTimerEngine(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取定时器接口 | |
| | 参数 | const IID & Guid　　接口标记<br>DWORD dwQueryVer　　接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

virtual void * __cdecl GetDataBaseEngine(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取数据库引擎接口 | |
| | 参数 | const IID & Guid　　接口标记<br>DWORD dwQueryVer　　接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

virtual void * __cdecl GetAttemperEngine(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取调度引擎接口 | |
| | 参数 | const IID & Guid　　接口标记 |

| | | |
|---|---|---|
| | | DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

virtual void * __cdecl GetTCPSocketEngine(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取网络引擎接口 | |
| | 参数 | const IID & Guid 接口标记<br>DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

virtual void * __cdecl GetAsynchronismEngine(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取异步引擎接口 | |
| | 参数 | const IID & Guid 接口标记<br>DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

virtual void * __cdecl GetDataBaseQueueService(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取数据库引擎数据队列接口 | |
| | 参数 | const IID & Guid 接口标记<br>DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

virtual void * __cdecl GetAttemperQueueService(const IID & Guid, DWORD dwQueryVer)

| | | |
|---|---|---|
| 说明 | 获取调度引擎数据队列接口 | |
| | 参数 | const IID & Guid 接口标记<br>DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

virtual bool __cdecl IsService()

| | | |
|---|---|---|
| 说明 | 判断服务状态 | |
| | 参数 | |
| | 返回 | 运行中返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl ControlService(void * pBuffer, WORD wDataSize)

| | | |
|---|---|---|
| 说明 | 外部控制 | |
| | 参数 | void * pBuffer 数据指针<br>WORD wDataSize 数据大小 |
| | 返回 | 运行中返回 ture,否则 false |
| 举例 | | |
| | | |

- **类设计**

| CServiceEngine | |
|---|---|
| 服务引擎 | |
| 模块 | ServiceEngine |
| 继承 | CServiceEngine → IServiceEngine → IUnknownEx |
| 实现 | IServiceEngine, IUnknownEx |

数据成员

| 名称 | 说明 |
|---|---|
| bool m_bService; | 运行标志 |
| CTimerEngine m_TimerEngine; | 定时器引擎 |
| CDataBaseEngine m_DataBaseEngine; | 数据库引擎 |
| CAttemperEngine m_AttemperEngine; | 调度引擎 |
| CTCPSocketEngine m_TCPSocketEngine; | 网络引擎 |
| CAsynchronismEnginem_AsynchronismEngine; | 异步引擎 |

方法

virtual bool __cdecl StartService(BYTE cbDBThreadCount = 12,
BYTE cbAttemperThreadCount = 1)

| | | |
|---|---|---|
| 说明 | 启动服务 | |
| | 参数 | BYTE cbDBThreadCount 数据库线程数量<br>BYTE cbAttemperThreadCount 调度引擎线程数量 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl StopService()

| | | |
|---|---|---|
| 说明 | 停止服务 | |
| | 参数 | |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetEventService(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置事件服务接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 事件服务接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetDataBaseSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置数据库钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 数据库钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

virtual bool __cdecl SetAttemperEngineSink(IUnknownEx * pIUnknownEx)

| | | |
|---|---|---|
| 说明 | 设置调度钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx 调度钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

| virtual bool __cdecl RegisterAsynchronismEngineSink(IUnknownEx * pIUnknownEx) | | |
|---|---|---|
| 说明 | 注册异步处理钩子接口 | |
| | 参数 | IUnknownEx * pIUnknownEx　　异步调用钩子接口指针 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

| virtual bool __cdecl InitServiceEngine(WORD wListenPort, WORD wMaxSocketItem) | | |
|---|---|---|
| 说明 | 设置网络引擎接口 | |
| | 参数 | WORD wListenPort　　监听端口<br>WORD wMaxSocketItem　　最大连接数目 |
| | 返回 | 操作成功返回 ture,否则 false |
| 举例 | | |

| virtual void * __cdecl GetTimerEngine(const IID & Guid, DWORD dwQueryVer) | | |
|---|---|---|
| 说明 | 获取定时器接口 | |
| | 参数 | const IID & Guid　　接口标记<br>DWORD dwQueryVer　　接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

| virtual void * __cdecl GetDataBaseEngine(const IID & Guid, DWORD dwQueryVer) | | |
|---|---|---|
| 说明 | 获取数据库引擎接口 | |
| | 参数 | const IID & Guid　　接口标记<br>DWORD dwQueryVer　　接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

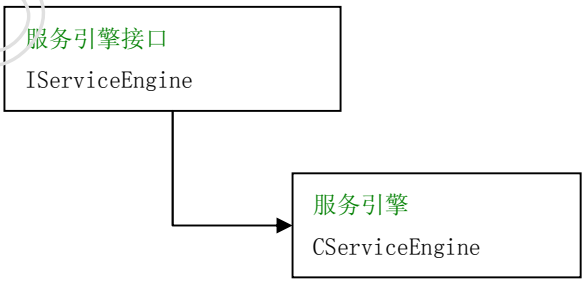
| virtual void * __cdecl GetAttemperEngine(const IID & Guid, DWORD dwQueryVer) | | |
|---|---|---|
| 说明 | 获取调度引擎接口 | |
| | 参数 | const IID & Guid　　接口标记<br>DWORD dwQueryVer　　接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

| virtual void * __cdecl GetTCPSocketEngine(const IID & Guid, DWORD dwQueryVer) | | |
|---|---|---|
| 说明 | 获取网络引擎接口 | |
| | 参数 | const IID & Guid　　接口标记<br>DWORD dwQueryVer　　接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |

| virtual void * __cdecl GetAsynchronismEngine(const IID & Guid, DWORD dwQueryVer) | | |
|---|---|---|
| 说明 | 获取异步引擎接口 | |
| | 参数 | const IID & Guid　　接口标记<br>DWORD dwQueryVer　　接口版本 |
| | 返回 | 操作成功返回接口指针 |

| | | |
|---|---|---|
| 举例 | | |
| virtual void * __cdecl GetDataBaseQueueService(const IID & Guid, DWORD dwQueryVer) | | |
| 说明 | 获取数据库引擎数据队列接口 | |
| | 参数 | const IID & Guid 接口标记<br>DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |
| virtual void * __cdecl GetAttemperQueueService(const IID & Guid, DWORD dwQueryVer) | | |
| 说明 | 获取调度引擎数据队列接口 | |
| | 参数 | const IID & Guid 接口标记<br>DWORD dwQueryVer 接口版本 |
| | 返回 | 操作成功返回接口指针 |
| 举例 | | |
| virtual bool __cdecl IsService() | | |
| 说明 | 判断服务状态 | |
| | 参数 | |
| | 返回 | 运行中返回 ture,否则 false |
| 举例 | | |
| virtual bool __cdecl ControlService(void * pBuffer, WORD wDataSize) | | |
| 说明 | 外部控制 | |
| | 参数 | void * pBuffer 数据指针<br>WORD wDataSize 数据大小 |
| | 返回 | 运行中返回 ture,否则 false |
| 举例 | | |
| | | |

- 类继承图




### 3.7.3 运行流程