**中期信息技术服务有限公司**

风控服务器

（RiskUserServer）

《概要设计说明书（软件）》

|  |  |  |
| --- | --- | --- |
| 文件状态：  √草稿   正式发布   正在修改 | 文件标识： |  |
| 当前版本： | V1.00 |
| 作 者： | 欧阳超伟 |
| 完成日期： | 2012年11月7日 |
| 审 定： |  |

版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V1.00 | 欧阳超伟 |  | 2012-10-7 |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

[1 引言 3](#_Toc340826844)

[1.1 编写目的 3](#_Toc340826845)

[1.2 背景 3](#_Toc340826846)

[1.3 定义 3](#_Toc340826847)

[1.4 参考资料 3](#_Toc340826848)

[2 任务概述 4](#_Toc340826849)

[2.1 目标 4](#_Toc340826850)

[2.2 运行环境 4](#_Toc340826851)

[2.3 需求概述 4](#_Toc340826852)

[3 总体设计 5](#_Toc340826853)

[3.1 基本设计 5](#_Toc340826854)

[3.2 基本结构和处理流程 6](#_Toc340826855)

[3.3 总体结构和模块划分 8](#_Toc340826856)

[3.4 配置设计 8](#_Toc340826857)

[3.5 尚未解决的问题 8](#_Toc340826858)

[4 接口设计 9](#_Toc340826859)

[4.1 与CTP风控和行情接口 9](#_Toc340826860)

[4.2 与风控客户端接口 9](#_Toc340826861)

[4.3 与管理客户端 9](#_Toc340826862)

[5 数据结构和数据库设计 9](#_Toc340826863)

[5.1 CDataCenter结构 9](#_Toc340826864)

[5.2 数据库结构 10](#_Toc340826865)

[5.3 管理客户端命令字和数据结构体 10](#_Toc340826866)

[5.4 风控客户端命令字和数据结构体 10](#_Toc340826867)

[6 系统出错处理设计 10](#_Toc340826868)

[6.1 出错信息 10](#_Toc340826869)

# 引言

## 编写目的

根据《中期统一风控平台需求》、《中期统一风控系统概要设计》和《数据描述》要求设计一个一个开放、分布、高可靠性、高效率、高兼容性的风控平台。对资管单位的帐户、人员、理财产品、资金情况、持仓明细情况、日内实时交易监视和报单审核等

## 背景

[待开发软件名称] ：风控服务器

[任务提出者或项目名称] ：

[设计者] ：欧阳超伟

[开发者] ：C++项目组

[目标用户] ：开发测试人员

[版本] ：v1.00

## 定义

* CTP：综合交易平台的简称，网址：<http://www.sfit.com.cn/>
* 风控接口：专指CTP风控接口，也就是FtdcRiskUserApi对应的控制功能
* 行情接口：专指CTP行情接口，也就是FtdcRiskUserApi对应的行情功能

## 参考资料

* + 1. 国家相关业务各种法律法规
    2. 交易所各交易、交割、结算、风险管理等规则；保证金监控中心各项规定
    3. 综合交易平台各种技术说明书
    4. 《中期统一风控平台需求》、《中期统一风控系统概要设计》和《数据描述》
    5. 中期集团其他风控业务需求

# 任务概述

## 目标

开发一个开放、分布、高可靠性、高效率、高兼容性的风控平台服务器。监控大区资管中心的交易风控，实时获取交易信息，总部实时体现各大区的风险情况，有具体的风险类型划分和风险等级，当出现风险事件后可以及时通知和下强制平仓指令。

## 运行环境

软件：皆由C++编写

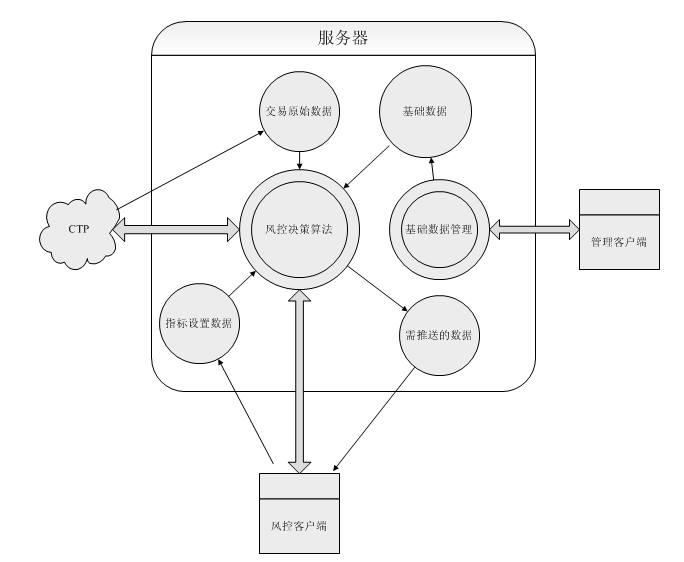
硬件：装载微软Windows的服务器

## 需求概述

1. 获取实时账号资金信息、持仓明细、报单、成交信息
2. 获取实时行情信息
3. 实现账号的分级管理
4. 实现风险算法和阀值的管理
5. 根据风险算法计算风控指标，触发风控事件
6. 实现风险事件的处理
7. 实现原始数据、基础管理数据和各种计算数据的数据备份

# 总体设计

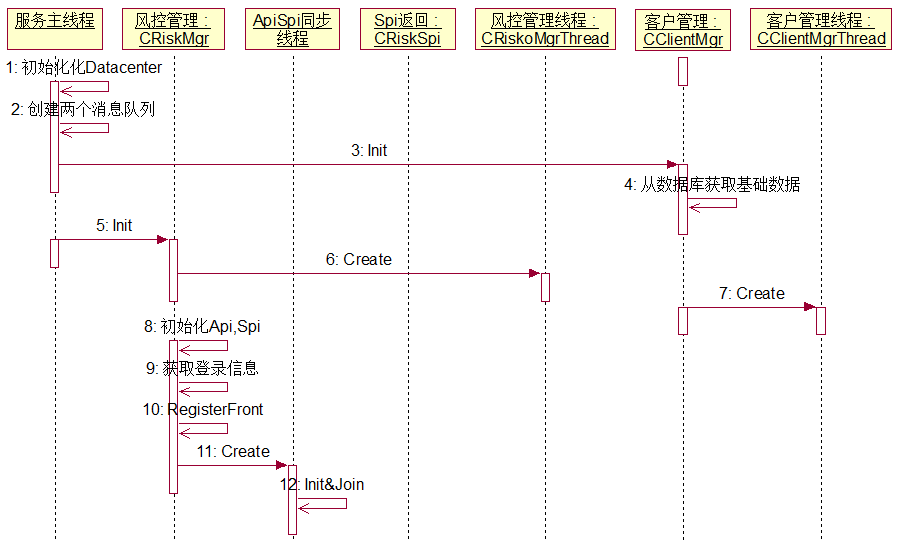
## 基本设计



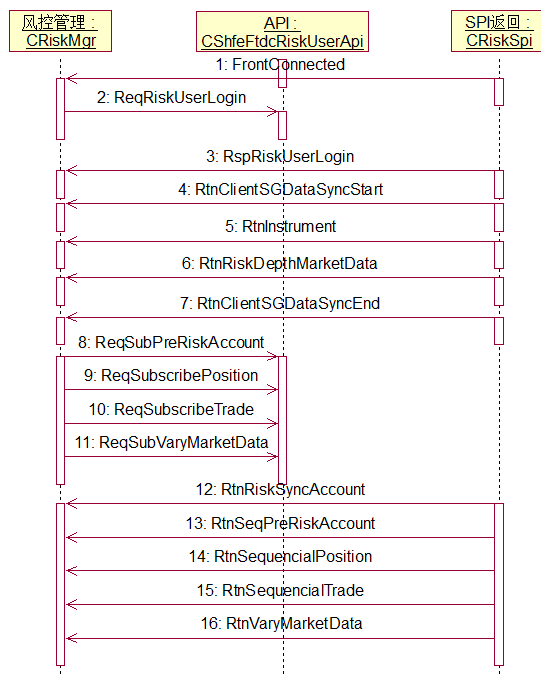
## 基本结构和处理流程



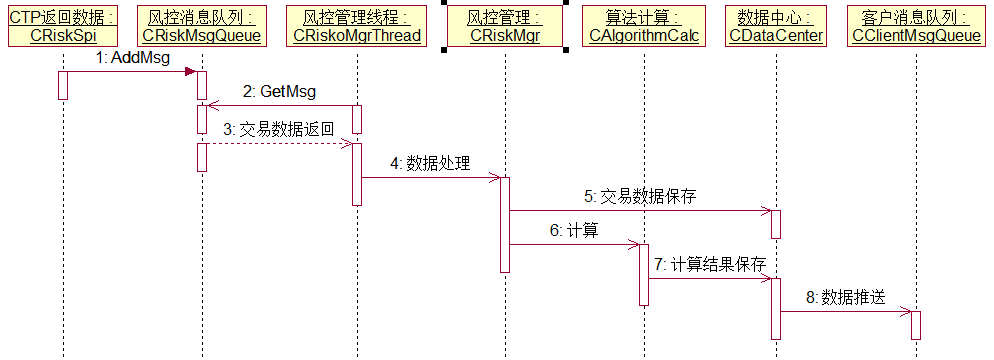
结构类图



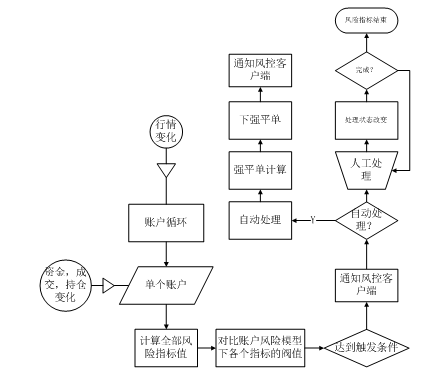
初始化流程图



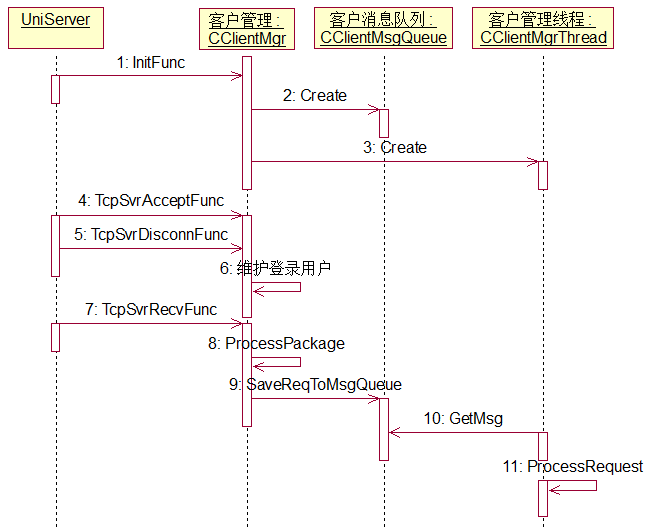
登录CTP风控和请求订阅数据



风控管理工作流程



风控计算流程



客户管理流程

## 总体结构和模块划分

| 模块名称 | 功能简述 |
| --- | --- |
| 风控、行情接口模块 | CRiskSpi，封装API和SPI，转发消息给CAlgoMsgQueue |
| 风控管理模块 | CRiskMgr，计算更新数据中心数据 |
| 算法计算模块 | CAlgorithmCalc包括风控消息管理、风控事件管理和风控策略计算 |
| 数据中心模块 | CDataCenter存储交易原始数据、基础数据、指标数据 |
| 客户端管理模块 | CClientMgr与风控客户端、管理客户端逻辑交互模块 |
| 数据库接口 | CDBMgr与Oralce或其他数据库打交道。 |
| 主框架及网络通信模块 | UniServer与风控客户端、管理客户端网络通信模块 |

## 配置设计

使用INI文件保存配置。

## 尚未解决的问题

* 网络不通畅时，对整个程序效能和表现的影响
* 多线程工作环境下，一个回调函数发生执行异常对整个程序的影响

# 接口设计

服务器连接CTP风控和行情接口，向风控客户端，管理客户端提供数据支持

## 与CTP风控和行情接口

*  创建风控API，连接登陆，CTP服务器
*  查询投资者资金，持仓，报单等交易原始数据
*  强平单录入
*  订阅，退订实时行情

## 与风控客户端接口

*  风控员登陆和权限管理
*  交易原始数据的请求和查询
*  消息的推送和显示
*  指标设置数据的发送和管理

## 与管理客户端

*  总部，大区管理员和风控员的登陆和权限管理
*  资管组织信息的录入和修改
*  理财产品的信息录入和修改

# 数据结构和数据库设计

## CDataCenter结构

* 风控资金账户昨日信息

map<string,CSeqPreRiskAccountFieldEx>

* 风控同步投资者资金账户

map<string,CRiskSyncAccountFieldEx>

* 持仓

map<string,map<int,CSequencialPositionFieldEx>>;

* 交易

map<string,map<int,CSequencialTradeFieldEx>>;

* 合约

map<string,CInstrumentFieldEx>;

* 行情

map<string,CDepthMarketDataFieldEx>;

* 委托交易账户和风险模型的关系

map< string,int>

* 账户风险指标值

map< string, map<int,double>>

* 风控预警设置

map<int, map<int, map<int,RiskLevel>>>

上面各个数据结构具体数据项详见《交易数据描述》，其他基础数据直接从数据库查询得到，内存中不缓存。

## CDataCenter接口

void AddAllInstrument(const CInstrumentFieldEx&);

void AddMarketData(const CDepthMarketDataFieldEx&);

void AddRiskFundAccount(const CRiskSyncAccountFieldEx&);

void AddPreRiskFundAccount(const CSeqPreRiskAccountFieldEx&);

void AddPosition(const CSequencialPositionFieldEx&);

void AddTrade(const CSequencialTradeFieldEx&);

void AddOrder(const CSequencialOrderFieldEx&);

void GetUserLoginField(CShfeFtdcReqRiskUserLoginField&) const;

void GetrFrontAddr(std::vector<std::string>&) const;

void GetAccountList( std::vector<std::string>&) const;

double GetTotalPostion(const std::string) const;

int GetInstrumentPostion(const std::string,const std::string) const;

RiskLevelType GetRiskLevel(const std::string,RiskIndicatorType,double) const;

## 数据库结构

详见数据库设计文档

## 管理客户端命令字和数据结构体

详见RiskManageCmd.h和RiskManageStruct.h

## 风控客户端命令字和数据结构体

详见风控客户端设计文档

# 强平算法设计

## 强平方案和规则数据说明

* 资管组织结构体中包含默认强平方案ID，用于自动强平
* 一个资管组织包含多个强平方案
* 一个强平方案包含多个强平规则
* 强平方案结构体

struct ForceCloseSolution

{

int nSolutionID;//强平方案ID

int szSolutionName;//强平方案名称

int nAssetMgmtOrgID//所属资管中心ID

};

* 强平规则结构体

struct ForceCloseRuleField

{

int ForceCloseRuleID; ///强平规则ID

int nSolutionID; ///所属强平方案ID

TShfeFtdcInstrumentIDType InstrumentID; ///合约代码

TShfeFtdcBrokerIDType BrokerID; ///经纪公司代码

TShfeFtdcInvestorIDType InvestorID; ///投资者代码

TShfeFtdcPosiDirectionType PosiDirection; ///持仓多空方向

TShfeFtdcHedgeFlagType HedgeFlag; ///投机套保标志

TShfeFtdcPositionDateType PositionDate; ///持仓日期

TShfeFtdcVolumeType Position; ///当前持仓数量

TShfeFtdcVolumeType FCPosition; ///需强平持仓数量

TShfeFtdcForceClosePriceTypeType FCPriceType; ///强平价格类型

TShfeFtdcVolumeType PriceTick; ///限价调整点数

TShfeFtdcPriceType FCPrice; ///强平价格

TShfeFtdcMoneyType ReleaseMargin; ///平仓释放的保证金

TShfeFtdcMoneyType CloseProfit; ///平仓盈亏

TShfeFtdcMoneyType ExchReleaseMargin; ///平仓释放的交易所保证金

};

///强平应答

struct RspForceClosePositionField

{

TShfeFtdcInstrumentIDType InstrumentID; ///合约代码

TShfeFtdcBrokerIDType BrokerID; ///经纪公司代码

TShfeFtdcInvestorIDType InvestorID; ///投资者代码

TShfeFtdcPosiDirectionType PosiDirection; ///持仓多空方向

TShfeFtdcHedgeFlagType HedgeFlag; ///投机套保标志

TShfeFtdcPositionDateType PositionDate; ///持仓日期

TShfeFtdcVolumeType Position; ///当前持仓数量

TShfeFtdcVolumeType FCPosition; ///需强平持仓数量

TShfeFtdcForceClosePriceTypeType FCPriceType; ///强平价格类型

TShfeFtdcVolumeType PriceTick; ///限价调整点数

TShfeFtdcPriceType FCPrice; ///强平价格

TShfeFtdcMoneyType ReleaseMargin; ///平仓释放的保证金

TShfeFtdcMoneyType CloseProfit; ///平仓盈亏

TShfeFtdcForceCloseSceneIdType FCID; ///强平场景编号

TShfeFtdcTimeType Time; ///辅助强平单的生成时间

TShfeFtdcMillisecType CurrMillisec; ///当前时间（毫秒）

TShfeFtdcMoneyType ExchReleaseMargin; ///平仓释放的交易所保证金

};

///风控强平报单输入

struct ForceCloseOrderField

{

TShfeFtdcForceCloseTypeType FCType; ///风控强平类型

TShfeFtdcTimeType Time1; ///辅助强平单的生成时间

TShfeFtdcMillisecType Millisec1; ///辅助强平单的生成时间（毫秒）

TShfeFtdcTimeType Time2; ///强平单的提交时间

TShfeFtdcMillisecType Millisec2; ///强平单的提交时间（毫秒）

TShfeFtdcForceCloseSceneIdType FCSceneId; ///强平场景编号

TShfeFtdcBrokerIDType BrokerID; ///经纪公司代码

TShfeFtdcInvestorIDType InvestorID; ///投资者代码

TShfeFtdcInstrumentIDType InstrumentID; ///合约代码

TShfeFtdcOrderRefType OrderRef; ///报单引用

TShfeFtdcUserIDType UserID; ///用户代码

TShfeFtdcOrderPriceTypeType OrderPriceType; ///报单价格条件

TShfeFtdcDirectionType Direction; ///买卖方向

TShfeFtdcCombOffsetFlagType CombOffsetFlag; ///组合开平标志

TShfeFtdcCombHedgeFlagType CombHedgeFlag; ///组合投机套保标志

TShfeFtdcPriceType LimitPrice; ///价格

TShfeFtdcVolumeType VolumeTotalOriginal; ///数量

TShfeFtdcTimeConditionType TimeCondition; ///有效期类型

TShfeFtdcDateType GTDDate; ///GTD日期

TShfeFtdcVolumeConditionType VolumeCondition; ///成交量类型

TShfeFtdcVolumeType MinVolume; ///最小成交量

TShfeFtdcContingentConditionType ContingentCondition;///触发条件

TShfeFtdcPriceType StopPrice; ///止损价

TShfeFtdcForceCloseReasonType ForceCloseReason; ///强平原因

TShfeFtdcBoolType IsAutoSuspend; ///自动挂起标志

TShfeFtdcBusinessUnitType BusinessUnit; ///业务单元

TShfeFtdcRequestIDType RequestID; ///请求编号

TShfeFtdcBoolType UserForceClose; ///用户强评标志

TShfeFtdcFrontIDType FrontID; ///前置编号

TShfeFtdcSessionIDType SessionID; ///会话编号

}

## 强平方案和规则数据管理

* 请求强平方案详情，参数：资管ID
* 请求强平规则详情，参数：方案ID
* 请求删除强平方案，参数：方案ID
* 请求修改方案名称，参数：新名称字符串
* 请求修改默认强平方案，参数：资管ID和默认强平方案ID
* 请求添加强平方案，参数：方案详情
* 请求删除强平规则，参数：强平规则ID
* 请求修改强平规则，参数：强平规则详情
* 请求添加强平规则，参数：强平规则详情

## 强平计算

* 请求强平计算，参数：强平账号，强平方案ID，强平算法。
* 强平计算返回，参数：多个强平应答结构体

## 强平下单

* 请求强平报单录入，参数ForceCloseOrderField
* 强平报单录入返回

# 系统出错处理设计

## 出错信息

定义全部错误类型信息于一个文件，全部与风控客户端、管理客户端接口错误时都返回一个错误码，错误码可以获取错误描述文字。