**【项目名称】**

**概要设计说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号： | V1.0 | 文档编号： | NG\_TS\_CFDB\_SDS |
| 文档密级： | 内部公开 | 归属部门/项目： | 专业系统部 |
| 编写人： | 郭淦 | 生效日期： | 2018-02-26 |

**版权信息**

本文件涉及之信息，属南京轨道交通系统工程有限公司所有。

未经南京轨道交通系统工程有限公司允许，文件中的任何部分都不能以任何形式向第三方散发。

网址：http://www.nanjingrail.com/

**文档修订记录**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订人** | **修订说明** | **修订状态** | **审核日期** | **审核人** | **批准人** |
| V1.0 | 2018-02-26 | 郭淦 | 草稿版 | A |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

修订状态：A--增加，M--修改，D--删除

日期格式：YYYY-MM-DD

**目　录**

[1. 引言 1](#_Toc287377107)

[1.1. 编写目的 1](#_Toc287377108)

[1.2. 背景 1](#_Toc287377109)

[1.3. 术语 1](#_Toc287377110)

[1.4. 预期读者与阅读建议 1](#_Toc287377111)

[1.5. 参考资料 1](#_Toc287377112)

[2. 总体设计 2](#_Toc287377113)

[2.1. 设计概述 2](#_Toc287377114)

[2.1.1 设计约束 2](#_Toc287377115)

[2.1.2 设计策略 2](#_Toc287377116)

[2.1.3 设计实现 2](#_Toc287377117)

[2.2. 设计目标 2](#_Toc287377118)

[2.3. 运行环境 2](#_Toc287377119)

[2.4. 平台架构 2](#_Toc287377120)

[2.5. 总体设计思路和处理流程 3](#_Toc287377121)

[2.6. 模块结构设计 3](#_Toc287377122)

[2.7. 功能需求与程序模块的关系（可选） 3](#_Toc287377123)

[2.8. 尚未解决的问题 3](#_Toc287377124)

[3. 接口设计 3](#_Toc287377125)

[3.1. 用户接口 3](#_Toc287377126)

[3.2. 外部接口 4](#_Toc287377127)

[3.3. 内部接口 4](#_Toc287377128)

[4. 界面总体设计 4](#_Toc287377129)

[5. 数据结构设计 4](#_Toc287377130)

[5.1. 设计原则 4](#_Toc287377131)

[5.2. 数据库环境说明 4](#_Toc287377132)

[5.3. 数据库命名规则 4](#_Toc287377133)

[5.4. 逻辑结构 4](#_Toc287377134)

[5.5. 物理存储 4](#_Toc287377135)

[5.6. 数据备份和恢复 4](#_Toc287377136)

[6. 系统出错处理设计[可选] 4](#_Toc287377137)

[6.1. 出错信息 4](#_Toc287377138)

[6.2. 补救措施 5](#_Toc287377139)

[6.3. 系统维护设计 5](#_Toc287377140)

[7. 系统安全设计 5](#_Toc287377141)

[7.1. 数据传输安全性设计 5](#_Toc287377142)

[7.2. 应用系统安全性设计 5](#_Toc287377143)

[7.3. 数据存储安全性设计 5](#_Toc287377144)

[8. 系统部署（可选） 6](#_Toc287377145)

# 引言

* 1. 编写目的

本文为地铁ISCS系统配置库子系统设计说明书，本部分与ISCS系统总体设计和其他各子系统设计共同组成ISCS概要设计文件。

系统的接口设计遵守《地铁ISCS系统技术规范》的规定，此技术标准与本概要设计作为软件详细设计和程序编制的设计依据。

* 1. 背景

ISCS系统负责地铁一条线路上的全部设备和系统监视和控制，ISCS系统需要维护大概30~100万个监视和控制数据点。这些点的状态发生变化，ISCS系统能够及时把状态的变化呈现在监视画面上；监视终端给设备或系统发送控制命令，ISCS系统能够及时把控制下发给对应的设备或系统。

ISCS系统使用配置库服务CFDB来管理维护配置数据库的表结构和提供统一的API层给调用服务使用（如RTDB、配置工具等），充分利用关系数据库的特性，来存储和维护整个ISCS系统的配置数据。

CFDB服务作为可以独立部署运行的程序存在。

* 1. 术语

术语和缩写参照《综合监控（实时数据库）软件需求规格说明书》。

* 1. 预期读者与阅读建议

描述本文档的主要读者，以及这些读者在阅读时的阅读重点与建议。可用列表的方式列出。如表：

|  |  |
| --- | --- |
| 预期读者 | 阅读重点 |
| 需求发起者和需求调研人员 | 系统架构和功能模块是否满足需求 |
| 项目负责人 | 系统架构设计是否合理，功能模块划分是否明确 |
| 系统设计人员 | 系统架构设计、功能模块划分是否合理 |
| 代码编写人员 | 根据系统架构、功能模块设计编码实现 |
| 系统测试人员 | 根据模块功能进行测试 |
|  |  |

* 1. 参考资料

N/A。

# 总体设计

* 1. 设计概述

### 2.1.1 设计约束

本部分的设计必须以《综合监控软件需求规格说明书》定义的需求为依据，按照需求文档的要求设计CFDB服务的系统架构和定义系统的功能模块。如果设计的功能是需求文档未做要求的情况，该功能不能影响或改变需求定义的功能。

### 2.1.2 设计策略

系统架构设计按照功能和业务逻辑关系进行模块化划分，不同的模块实现不同的功能，不允许跨模块的功能。模块与模块要低耦合高内聚，模块与模块通过接口交换数据。

### 2.1.3 设计实现

CFDB系统主要分为SDK接口模块、Tcp服务模块、数据库模块、缓存模块等。总体流程设计如图：



CFDB配置服务实现统一的操作数据库的方法。提供不同需求的数据返回，分别是对RTDB的tcp协议的数据形式，以及对工具等服务的http协议的数据形式。

* 1. 设计目标

CFDB配置库支持设备建模、用户权限配置、统计和报警等定义配置，为ISCS的其他业务系统提供对配置数据库统一的操作接口。

CFDB配置库支持设备建模，即CFDB提供新建设备模板和通过设备模板新增设备功能。

同时也提供对用户权限的对功能、角色、用户、用户组的定义和权限分配。也提供对统计、报警的定义以及公式的定义。

CFDB配置库也提供了对配置数据库的统一操作接口。不仅能支持也实时库的数据交换，也支持对界面工具类的数据交换。CFDB为ISCS的业务系统、服务、HMI等子系统提供高效的实时数据查询。CFDB支持支持大量客户端的并发查询请求，并及时对请求作出处理和响应。

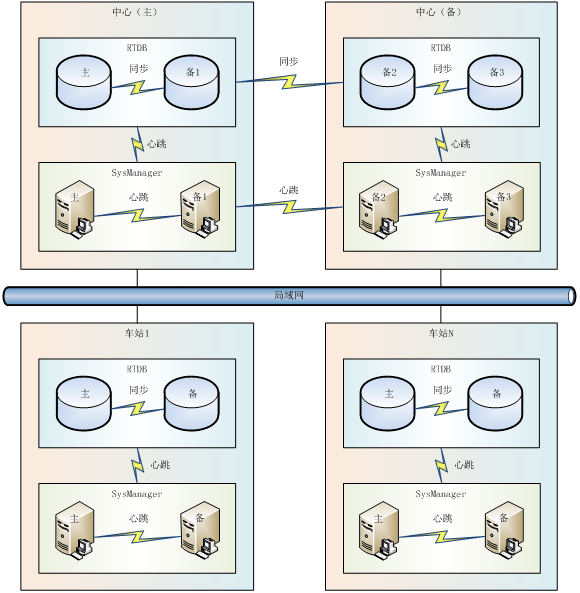
* 1. 运行环境

OS版本。实时数据库支持的OS版本有RHEL7\_x64、CentOS7\_x64、Ubuntu\_Server14\_x64、Linux主流发行版x64（内核2.8及以上）。

硬件要求。处理器8核（及以上），单核频率3.0 GHZ（及以上），内存32 GB（及以上），磁盘1 TB（及以上），1000M以太网卡，1000M网络带宽。

软件要求。中心车站提供Oracle数据库，站点车站提供PostgreSQL数据库。

* 1. 平台架构



* 1. 总体设计思路和处理流程

CFDB配置服务的业务流程设计为两个主要部分：配置库数据更新和配置库数据获取。以保证配置的数据安全入库、最快的响应请求数据为优先级最高的准则，配置服务的所有功能和模块的设计、划分都以此为原则。

配置库数据更新流程如下图：



1. 请求解析

配置库提供不同方式的数据库操作请求。根据调用方式的不同，执行不同的数据库操作。方式一为调用方通过配置服务提供的SDK库接口调用，根据Tcp请求执行数据库数据的更新；方式二为调用方通过配置服务提供的Http接口调用，根据Http请求执行数据库数据的更新。

1. 参数封装

将从请求中获取的请求参数，封装成接口需要的参数形式。

1. 数据库接口调用

实现统一的数据库接口，数据更新时调用对应的更新方法，更新数据库数据。若未执行成功，返回出错信息，并进行日志记录。

1. 数据库同步

配置库数据更新之后，需要同步到备用站点配置库中。通过消息中间件MQ进行主备配置库数据同步。若执行失败，将需要更新的数据进行文件备份，调度模块会根据备份的文件，重新执行同步操作，并进行日志记录。

1. 更新缓存

配置库数据更新之后，还需要更新缓存。更新缓存和数据同步并行执行，但更新缓存优先级低于数据同步。若更新缓存失败，将需要更新的数据存入不同目录下的文件中进行备份，调度模块会根据备份的文件，重新执行更新缓存操作，并进行日志记录。

配置库数据获取流程如下图：



1)请求解析

配置库提供不同方式的配置库数据获取请求。通过解析模块，分别进行不同方式的获取配置库数据。

2）数据拆装

将请求中获取配置数据库的参数信息提取出来，并封装成统一接口需要的参数形式。

1. 缓存获取

若为Tcp请求的数据获取方式，只从缓存中获取。如果失败，返回错误信息并记录， 数据获取流程结束，不做任何其他处理。

4）数据库接口

若为Http请求方式，数据获取时调用相应的数据查询接口。若成功返回查询数据，失败则进行日志记录。

1. 数据返回

根据请求数据的方式不同，将返回的数据以不同的方式返回。

* 1. 制作购买或复用的分析

N/A

## 模块结构设计

按照实现的功能对CFDB配置服务进行模块划分和设计。CFDB服务划分为连接管理层、请求处理模块、请求数据处理模块、统一接口层、数据库操作模块、协议转换模块、缓存模块、同步模块、备份处理模块等功能模块。

 2.7.1 模块划分

1）统一API接口层

配置服务提供封装好的SDK接口层，以统一的接口提供给以高性能方式进行数据库操作 的调用者进行调用。

1. SDK接口层

SDK接口层对调用方提供统一的接口，同时与Tcp服务层建立连接，并进行数据库操作 请求，同时获取响应数据。

3）Tcp服务层

Tcp服务层维护与各个网络连接的通信状态，向其他模块提供接收的请求数据，同时向连接请求方提供响应数据。

4）数据解析层

根据接收的请求数据，进行数据的解析和封装，向其他模块提供封装好的数据信息。

5）请求处理层

根据封装好的请求数据进行判断处理，控制请求处理流程走向，是否通过缓存模块获取，以及返回数据协议的形式的判定。

6）数据库接口层

提供对配置库的数据更新和数据获取的接口集。

7）协议处理层

根据不同的协议请求，对请求的响应数据进行协议的转换。

8）缓存模块

为更快的对ISCS其他系统提供数据获取操作，CFDB配置服务提供了缓存模块。缓存模块只保存RTDB需要的数据。并且保证缓存的数据新鲜度能伴随着数据库的更新，能对外提供与配置数据库中同样的数据信息。

9）备份处理模块

备份处理模块是为了应对已通过网络获取到的配置库更新数据，且在数据库更新操作

和进行数据缓存时因为不同的原因更新缓存失败，所进行的补救操作。将更新的数据，在不同的目录文件中进行数据备份。备份的文件会被备份调度模块重新执行，以保证缓存数据与配置库数据的一致性。

2.7.2 模块关系

Tcp服务层详细设计如下图：

时序图如下：

1.SDK层接口

SDK接口在被调用时，首先会创建一个线程，在线程中创建Tcp协议的socket连 接，对请求进行解析并根据定义的Tcp协议向Tcp服务端分页请求数据，以满足可 能的高并发的批量数据请求。

2.Java服务层

1）Tcp服务层

Tcp服务层在接收到SDK的Tcp客户端请求时，会与Tcp客户端建立socket连接。

并根据接收的请求调用数据库接口获取配置库数据，同时根据定义的Tcp协议，将 数据通过socket返回给请求方。

2）数据库接口层

数据库接口层被Tcp服务层调用时，向数据库获取数据。并将数据返回到Tcp服务 层，以供Tcp服务层返回请求数据。

* 1. 功能需求与程序模块的关系（可选）

本条用一张如下的矩阵图说明各项功能需求的实现同各块程序模块的分配关系，本表可用需求跟踪距阵替代：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 程序1 | 程序2 | …… | 程序m |
| 功能需求1 | √ |  |  |  |
| 功能需求2 |  | √ |  |  |
| …… |  |  |  |  |
| 功能需求n |  | √ |  | √ |

* 1. 尚未解决的问题

说明在概要设计过程中尚未解决而设计者认为在系统完成之前必须解决的各个问题。

# 接口设计

* 1. 用户接口

说明与用户接口的输入输出关系，以及将向用户接口提供的命令和它们的语法结构，以及软件的回答信息。

* 1. 外部接口

说明本系统同外界的所有接口的安排包括软件与硬件之间的接口、本系统与各支持软件之间的接口关系、协议要求等。

### 3.2.1 接口

1. RTDB接口

数据装载

返回码 接口函数(资源描述符，域标识，回调函数(输出参数));

int async\_conf\_module\_load\_table(String tableName, int domaiId，void func(String tableName, void \*record)));

数据差异同步

返回码 接口函数(资源描述符，域标识，当前时间戳，回调函数(输出参数));

int conf\_module\_sync\_table(String tableName, int domaiId，long timestamp,void func(String tableName, void \*record)));

1. 通用接口

struct recordSet\* conf\_load\_table(char\* str\_sql);

说明：获取记录

参数：str\_sql - sql语句

返回：记录缓存指针

Struct recordSet{

Vector<char\* filed\_name>;//列名称列表

list<list<char\* value>>;//列记录列表

int sql\_result;//操作返回码

}

Char\* get\_table\_value(int index,char\* field,struct recordSet\* record，int\* code);

说明：获取指定记录

参数：

Index - 行索引

Field - 列名称

Record - 记录缓存指针

Code - 返回码

返回：指定行记录中列名称对应的记录值

Int get\_record\_size(struct recordSet\* record);

说明：获取记录总数量

参数：Record - 记录缓存指针

返回：记录总数量

Int record\_free(struct recordSet\* record);

说明：释放记录缓存

参数：Record - 记录缓存指针

返回：返回码

### 3.2.2 协议

1.请求包格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | 起始位 | 2Byte | 7F 7E | 固定值 |
| 包编号 | 8Byte(int) | 时间值 | 毫秒数 |
| 域编号 | 4Byte(int) |  |  |
| 操作编号 | 4Byte(int) |  |  |
| 操作描述长度 | 4Byte(int) |  |  |
| 操作描述(表名称) | nByte |  |  |
| 包体总长度 | 4Byte(int) |  | 包体和包尾之和 |
| 包体 | 片长度(可选) | nByte |  | 参数片为不定长的字符串，用此字段指明长度；否则不使用 |
| 参数片 | nByte |  |  |
| ... |  |  |  |
| 片n长度(可选) | 4Byte(int) |  |  |
| 参数片n | nByte |  |  |
| 包尾 | 校验码 | 2Byte |  | CRC校验码 |
| 结束位 | 2Byte | 0D 0A | 固定值 |

2.响应包格式

1）数据包

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | 起始位 | 2Byte | 7F 7E | 固定值 |
| 包编号 | 8Byte(int) | 时间值 | 毫秒数 |
| 域编号 | 4Byte(int) |  |  |
| 操作编号 | 4Byte(int) |  |  |
| 操作描述长度 | 4Byte(int) |  |  |
| 操作描述(表名称) | nByte |  |  |
| 包体总长度 | 4Byte(int) |  | 包体和包尾之和 |
| 包体 | 结束标识 | 1Byte |  | 是否结束通讯 |
| 完成标识 | 1Byte |  | 是否有下一页 |
| 对象个数 | 4Byte(int) |  |  |
| 对象大小 | 4Byte(int) |  | 属性片总大小 |
| 片类型 | 4Byte(int) |  |  |
| 片长度(可选) | 4Byte(int) |  | 属性片为不定长的字符串，用此字段指明长度；否则不使用 |
| 属性片 | nByte |  |  |
| 属性名称长度 | 4Byte(int) |  |  |
| 属性名称 | nByte |  |  |
| 下个属性标志 | 1Byte |  | 0x01 |
| ... |  |  |  |
| 片类型 | 4Byte(int) |  |  |
| 片n长度(可选) | 4Byte(int) |  | 属性片为不定长的字符串，用此字段，否则没有 |
| 属性片n | nByte |  |  |
| 属性名称长度 | 4Byte(int) |  |  |
| 属性名称 | nByte |  |  |
| 下个属性标志 | 1Byte |  | 0x00 |
| 包尾 | 校验码 | 2Byte |  | CRC校验码 |
| 结束位 | 2Byte | 0D 0A | 固定值 |

包头：包头分为起始位、包编号、表编号、包长度。

1. 起始位

起始位为固定值，以0x7F 0x7E开始。

1. 包编号

包编号为请求方发起请求时的时间值(毫秒数值)，接收方收到请求后所做的响应数 据包中必须以相同的包编号回应。

1. 操作编号

操作编号为所有数据库操作请求的字典索引值，每个编号对应相应的数据库操作。

接收方收到请求后所做的响应数据包中必须以相同的操作编号回应。

1. 包长度

包长度为包体和包尾的总长度。

包体：包体为实际数据的信息存储空间。

1. 消息片

消息片中存储的为对应数据库操作的sql语句，为变长的字符串。

1. 参数片

请求数据包中，参数片存储对应sql操作的参数值。

1. 数据片

响应数据包中，数据片存储对应的响应数据对象的属性值。

1. 片长度

对应消息片、参数片、数据片长度。

包尾：数据包的末尾存储空间。

1）校验码

固定2个字节长度，存储CRC校验码。

2）包尾

固定2个字节长度，且为固定值0x0D 0x0A。

数据包示例(示例用十六进制和字符串组合表示，实际数据包全为十六进制)：

备注：

以Person对象为例

Person{

String name;

Int age;

}

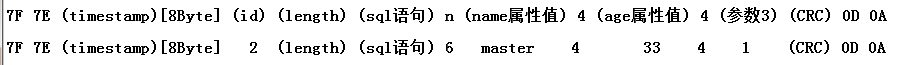
1. 新增

SQL语句：insert into t(name,age) values(:1,:2) [共37Byte]



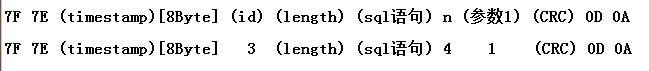
1. 更新

SQL语句：update t set name=:1,age=:2 where id=:3



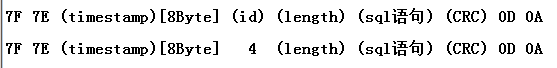
1. 删除

SQL语句：delete from t where id=:1

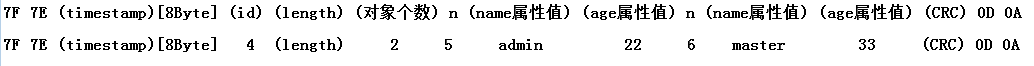


1. 查询

SQL语句：select \* from t



1. 响应数据包



1. 响应数据结束包



* 1. 内部接口

说明本系统之内的各个系统元素（各层模块、子程序、公用程序等）之间的接口的安排，包括设计用关系、输入输出要求、语法结构等。

# 界面总体设计

说明界面总体布局和风格设计。

# 数据结构设计

可以在本文说明也可以单独使用数据库设计说明书描述

* 1. 设计原则

给出系统数据库的设计原则。

* 1. 数据库环境说明

简单介绍一些数据库直接有关的支持软件，如数据库管理系统、存储定位程序和用于装入、生成、修改、更新数据库的程序等。说明这些软件的名称、版本号和主要功能特性。

* 1. 数据库命名规则

联系用途，详细说明用于唯一地标识该数据库的代码、名称或标识符，附加的描述性信息亦要给出。

* 1. 逻辑结构

数据库设计人员根据需求文档，利用数据建模技术来描述逻辑数据库结构。要求使用PowerDesigner或Visio创建数据库PDM模型。此处只需列出PDM模型名称。

* 1. 物理存储

描述整个逻辑数据模型是如何被转换为数据文件（物理模式）。文件结构类型在这里应清楚的体现。

* 1. 数据备份和恢复

描述数据库的备份和恢复策略。

# 系统出错处理设计[可选]

* 1. 出错信息

用一览表的方式说明出错的类型，以及每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法。例如：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **错误分类** | **子项及其编码** | **错误名称** | **错误代码** | **备注** |
| 数据库错误 | 连接 | 连接超时 | 100001001 |  |
| 连接断开 | 100001002 |  |
| 数据库本身错误代码 | 数据库本身错误代码 | 100002+数据库错误代码 |  |
| TCP连接错误 | 连接 | 连接超时 | 101001001 |  |
| 连接断开 | 101001002 |  |
| 其它TCP连接错误(socket自身错误代码) |  | 101002+ socket错误代码 |  |
| 配置信息错误 | 未配置输入参数 |  | 102001 |  |
| 未配置输出参数 |  | 102002 |  |

* 1. 补救措施

说明故障出现后可能采取的变通措施，包括：

a．后备技术 说明准备采用的后图示技术，当原始系统数据万一丢失时启用的副本的建立和启动的技术，例如周期性地把磁盘信息记录到磁带上去就是对于磁盘媒体的一种后备技术；

b．降效技术 说明准备采用的后备技术，使用另一个效率稍低的系统或方法来求得所需结果的某些部分，例如一个自动系统的降效技术可以是手工操作和数据的人工记录；

c.恢复及再启动技术 说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。

* 1. 系统维护设计

说明为了系统维护的方便而在程序内部设计中作出安排，包括在程序中专门安排用于系统的检查与维护的检测点和专用模块。

# 系统安全设计

* 1. 数据传输安全性设计

说明在数据通信和传输过程中安全性设计。

* 1. 应用系统安全性设计

说明在访问应用系统过程中用户以及访问权限、操作等安全性设计。

* 1. 数据存储安全性设计

说明在数据和文件在存储过程中的安全性设计。

# 系统部署（可选）

给出系统部署方案，尽量使用图表的形式，并辅以必要的文字说明。