**综合监控概要设计说明书**

**系统管理**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号： | V1.0 | 文档编号： | NG\_TS\_SMS\_SDS |
| 文档密级： | 内部公开 | 归属部门/项目： | 综合自动化部 |
| 编写人： | 刘涛 | 生效日期： | 2018-03-01 |

**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 | 2018-03-01 | 刘涛 | 正式版 | A |  |  |  |
| V1.1 | 2018-04-01 | 刘涛 | 正式版 | M |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

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](#_Toc507691798)

[1.1. 编写目的 1](#_Toc507691799)

[1.2. 背景 1](#_Toc507691800)

[1.3. 术语 1](#_Toc507691801)

[1.4. 预期读者与阅读建议 1](#_Toc507691802)

[1.5. 参考资料TBD 2](#_Toc507691803)

[2. 总体设计 2](#_Toc507691804)

[2.1. 设计概述 2](#_Toc507691805)

[2.1.1. 设计约束 2](#_Toc507691806)

[2.1.2. 设计策略 2](#_Toc507691807)

[2.1.3. 设计实现 3](#_Toc507691808)

[2.2. 设计目标 4](#_Toc507691809)

[2.3. 运行环境 4](#_Toc507691810)

[2.4. 平台架构 5](#_Toc507691811)

[2.5. 总体设计思路和处理流程 5](#_Toc507691812)

[2.6. 制作购买或复用的分析 5](#_Toc507691813)

[2.7. 模块结构设计 5](#_Toc507691814)

[2.7.1. 用户管理 5](#_Toc507691815)

[2.7.2. 主从互备 6](#_Toc507691816)

[2.7.3. 系统配置 6](#_Toc507691817)

[2.7.4. 安全管理 7](#_Toc507691818)

[2.7.5. 日志记录 7](#_Toc507691819)

[2.8. 功能需求与程序模块的关系（可选） 7](#_Toc507691820)

[2.9. 尚未解决的问题 7](#_Toc507691821)

[3. 接口设计 7](#_Toc507691822)

[3.1. 用户接口 7](#_Toc507691823)

[3.2. 外部接口 7](#_Toc507691824)

[3.3. 内部接口 7](#_Toc507691825)

[4. 界面总体设计 7](#_Toc507691826)

[5. 数据结构设计 7](#_Toc507691827)

[5.1. 设计原则 7](#_Toc507691828)

[5.2. 数据库环境说明 7](#_Toc507691829)

[5.3. 数据库命名规则 7](#_Toc507691830)

[5.4. 逻辑结构 8](#_Toc507691831)

[5.5. 物理存储 8](#_Toc507691832)

[5.6. 数据备份和恢复 8](#_Toc507691833)

[6. 系统出错处理设计[可选] 8](#_Toc507691834)

[6.1. 出错信息 8](#_Toc507691835)

[6.2. 补救措施 8](#_Toc507691836)

[6.3. 系统维护设计 9](#_Toc507691837)

[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。 9](#_Toc507691838)

[7. 系统安全设计 9](#_Toc507691839)

[7.1. 数据传输安全性设计 9](#_Toc507691840)

[7.2. 应用系统安全性设计 9](#_Toc507691841)

[7.3. 数据存储安全性设计 9](#_Toc507691842)

[8. 系统部署（可选） 9](#_Toc507691843)

# 引言

* 1. 编写目的

本概要设计主要描述的是在轨道交通综合监控系统系统管理模块的解决方案。保障系统实时高效、可靠、安全的运行。

* 1. 背景

作为大型分层分布式软件，系统管理相关功能必不可少。综合监控软件V2.0平台系统管理根据前期架构设计可以分为以下几个模块：

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | 模块名称 | 模块说明 | 其他说明 |
| 1 | 用户管理 | 用户定义；用户组定义；角色定义。 | 用来管理用户、用户组、密码、角色、角色与画面的关联关系，实现全局的权限管理 |
| 2 | 冗余管理 | 冗余管理：含守护进程自我裁定设计，被守护进程的主备切换设计。 | 需要和实时库、前置、HMI等各被管理服务定义冗余管理接口 |
| 3 | 进程管理 | 登记各进程的启动顺序、启动参数、进程状态等。 |  |
| 4 | 设备管理 | 综合监控系统内的服务器、工作站的运行工况进行监控，含服务器和工作站的CPU占用率、内存占用率等指标。 | 实时监控设备状态信息，发生超限时调用报警接口发送刚给实时库，产生报警 |
| 5 | 网络通断 | 对组成综合监控网络内各设备的网络通断情况进行监控。 | 实时监控各设备在线状态，获取工作站和服务器的网络服务状态 |
| 6 | 节点管理 | 查看本节点下运行的所有服务或者进程的名称、状态、启动时长等信息。  使用软件授权license管理服务器和工作站的接入合法性问题，每台服务器和工作站必须获得合法的license才能部署对应的服务端或者客户端。 | 暂时考虑使用软件授权的方式，管理综合监控系统合法性的问题。 |

考虑下各模块之间代码的组合方式（一个进程还是多个进程）

* 1. 术语

API：Application Programming Interface，应用程序接口。

SMRTDB:系统管理实时数据存储

* 1. 预期读者与阅读建议

|  |  |
| --- | --- |
| **预期读者** | **阅读建议** |
| 系统领导层 | 仔细阅读概述，编写目的，文档约定，系统功能介绍和维度指标说明。 |
| 业务部门、决策部门、具体的使用部门、业务员、系统管理员 | 仔细阅读文档约定，系统功能介绍和维度指标说明。  各个部门可重点阅读与本部门相关的内容。 |
| 参加需求评审的人员 | 仔细阅读全部内容。 |
| 系统设计人员 | 仔细阅读全部内容。 |
| 系统测试人员 | 仔细阅读文档约定，系统功能介绍和维度指标说明。 |

* 1. 参考资料TBD

列出有关的参考资料，如：

经过评审的《D17-09-综合监控软件平台v2.0系统需求规格说明书》

哈尔滨二号线综合监控系统用户需求规格书

# 总体设计

* 1. 设计概述

### 设计约束

功能需求约束：从整体上看，系统管理各模块的设计必须实现用户需求说明书中所要求的所有功能，当前设置的各模块认为已经是功能的具体划分。其实现方式则可由开发人员自由选择。

独立性和通用性约束：系统管理应当是一个相对独立的子系统，不应当向具体的应用层程序提出过多要求，不应当实现明显只针对特定应用的函数接口。整个系统管理必须是一个尽可能通用的平台，各种不同的应用程序能够在上面运行良好。系统管理提供通用的API或者接口供其他后台程序调用。

模块化约束：系统管理本身是一个比较大的子系统，为了方便开发和调试，系统管理必须有良好的模块化。模块之间尽量独立，模块之内功能尽量单一。本身系统管理内各个模块之间的功能相对独立，本身为系统的模块化设置提供了一个较好的基础。

考虑到基于Redis的存储设计，可以较好的实现数据的实时存储和处理，并且系统管理实时库的设计可以为大的实时库设计提供较好的参考经验，本项目在架构设计时考虑使用Redis搭建存储结构。各个模块在实时库中根据自身需要进行数据存储。

### 设计策略

扩展策略：因为系统管理是一个相对独立和通用的子系统，为了更好地实现通用性，各个功能模块和函数接口都应当是可扩展的。一些功能函数应当定义一些将来可能用到的参数，尽管这些参数所规定的功能在当前版本中不需要实现，但是必须考虑到以后的扩充。

考虑到当前人力和时间有限，在设计的时候可以将功能尽可能考虑清楚，但是设计明确的产品演进路线，首先要考虑的是实现基本的，现场急需的功能。

### 设计实现

本小节，描述系统管理内各模块的初步设计思路。

#### 2.1.3.1 用户管理

数据结构设计上使用用户、用户组、角色、功能（操作和浏览）、画面类型几个关键概念进行用户管理。明确各部分的概念，梳理各部分之间的继承关系。以上关系在商用数据库内配置。

在系统管理启动时，SMRTDB通过配置服务在商用库中加载用户的相关配置；

用户管理模块提供鉴权、权限查询接口给HMI，支持用户登录的密码鉴权以及画面权限加载工作；

* **元数据设计**

**用户模块**

用来定义用户的概念，即用户名，用户别名，用户ID,用户简单描述等信息，包含用户所属的组的信息。

**用户组：**

在本设计中用户组的概念相对重要，在继承角色的权限及画面关联关系后，可以通过定义用户组的位置，来进行角色的落地，即角色对应实际运营的部门。用户组的组长一般为班长或者部门领导。

每个用户组设置一个组长，组长在HMI界面上具有对本组成员的增删改查的功能。

用户组

一个用户组可以定义为新街口车站值班员，用户组的组长可设置为新街口站的站长。

**角色：**

在逻辑程度上来说是用户组的进一步提升，将拥有同一类操作权限及画面权限的用户组人员抽象为某一角色。

在角色下面可以定义该角色对应的操作权限、可以查看的画面的类型。

一个具体的角色定义可以为车站值班员或者维修工班人员。

**操作权限或者功能：**

操作权限包括浏览、遥控（可按责任区配置）、报警确认（可按责任区配置）等，认为是当前操作员对综合监控常见的操作的设置关系。

**画面类型：**

将某一专业的画面类型进行汇总和抽象，如布局图、PSCADA专业画面等。

通过角色将画面和权限的关系进行关联。

一个用户组可以具有多个角色；一个角色可以被多个用户组拥有；一个用户可以属于多个用户组。一个用户可以成为多个用户组的组长。

以上各定义的相互关系可以由下图表示：



图2-1 用户元数据相互关系

#### 2.1.3.2 冗余管理

主要功能流程设计如下：

1. 冗余管理启动，启动后冗余管理主备服务使用统一的机制进行主备判断。在主备正确判断后，行使冗余管理的职能。冗余管理主备服务之间保持主备状态报文通信，以确保在必要的时候进行系统管理服务主备的切换。

以上流程详见2.6.5.1小节设计

1. 被管理服务，如实时库、历史库、配置服务、前置服务以及HMI在启动后向系统管理进行服务注册，同一服务的主备进程根据注册的先后顺序进行裁决，即先到先得。
2. 被管理服务在运行期间和系统管理服务保持周期性通信，一旦发现自身主备状态被通知改变后，被管理服务在主进程内进行主备切换动作。

2、3点内容流程详见2.6.5.2小节设计

#### 2.1.3.2 进程管理

进程管理服务，用来对系统中所有的进程进行管理；进程管理提供API接口给各进程使用，其他各进程在启动时将注册信息按照约定好的格式发送给进程服务，进程服务根据注册信息进行进程的管理。没有注册的进程不在进程管理的范围内。

进程管理的内容主要包括以下几个：

1. 根据注册信息，获取各域当前运行的进程信息及其进程编号；
2. 周期对注册的进程进行查询，看对应的进程是否存在；
3. 已注册的进程消失，应根据拉起策略，执行对应的拉起动作；
4. 不能被拉起的进程，将其状态设置为故障，并通知告警服务。
   1. 设计目标
   2. 运行环境

|  |  |  |
| --- | --- | --- |
| **类别** | **标准配置** | **最低配置** |
| 计算机硬件 | TBD | Intel Xeon 双核CPU（64  位指令集兼容）1GB内存/40GB硬盘,两块相同的以太百兆网卡 |
| 软件 | TBD | CentOS 7 x64 |
| 网络通信 | TBD | TCP/IP协议栈支持，IP多播支持 |

* 1. 平台架构

**图 2系统架构**

* 1. 制作购买或复用的分析

系统管理使用开源Redis作为业务数据内存缓存，其他模块均自主开发，所有模块采用代码编译集成的形式复用。

* 1. 模块结构设计

### 用户管理

* 启动

基本的用户管理元数据已经在2.1.3.1设计中进行描述。用户管理正常运行的前提是配置数据的正常下载。当前设计默认的用户配置使用配置服务进行数据下载。配置服务保证中心和车站配置的同步。

* 用户登录鉴权管理

HMI界面发起登录请求后，具体的鉴权应在用户管理后台进行；界面将对应的用户、登录地点（终端主机名）、密码等信息发送到鉴权模块，由鉴权模块进行鉴权操作。

所有的用户的相关信息应该在控制中心节点进行配置，配置时可以选择用户所属域，在配置结束后，对应的配置信息应同步到各个车站的配置数据库中。

SMRTDB在启动时在控制中心按域进行配置信息下装，当控制中心与本地网络中断后，SMRTDB应自动从本地配置库同步数据。

至于判断流程，用户登录鉴权时，应首先判断当前登录用户是否有在本地登录的权限，再判断是否已在其他终端登录（目前暂时的策略为不允许用户在不同的客户端登录，后期可以考虑进行适当的扩展），如果以上条件都判断成功后，再判断登录密码是否正确。以上流程可由下图表示：



**图 4用户鉴权流程示意图**

* 用户登录状态管理

记录用户登录状态，主要包括登录名，登录开始时间，登录失效时间，登录IP地址。

* 用户权限查询

用户管理为HMI提供查询接口，可根据HMI的要求返回指定用户可以做的操作权限，关联的画面类型和默认的屏幕画面配置情况。

### 节点管理

节点应该定义全线所有的管理节点，从大的层面上来说分为中心和车站，具体到每一个点可以分为综合监控服务器和综合监控工作站。

节点管理应定义全线所有节点的主机名、IP地址、主备网卡数量、主备IP地址等内容，详情参照系统管理节点管理部分内容。

使用软件授权license管理服务器和工作站的接入合法性问题，每台服务器和工作站必须获得合法的license才能部署对应的服务端或者客户端。

### 设备管理

设备管理主要监控设备本身的CPU占用率、内存占用率等指标，调用操作系统相关接口查询相关指标，在超限时能够调用报警服务进行报警生成。

### 网络通断管理

在设备管理运行的后台不断尝试对网关、各工作站以及硬件前置FEP进行网络连接尝试，当发现与某一设备连接掉线时，则生成相应的报警。该项功能后续可以考虑与NMS系统合并，前期不建设网管系统的前提下，可先建设该功能。

### 冗余管理

冗余管理进程应作为系统管理的主要进程，重点关注；冗余管理进程应在系统管理进程第一位启动；冗余管理进程启动后，自己裁决自身系统管理主备信息。被管理进程启动后，向冗余管理进程发送信号，进项主备注册，默认同一服务先注册的实例被设置为主进程。

当冗余管理进程发现被管理进程的主进程实例无法正常响应时，应向被管理进程的备用进程发送消息，通知其升级为主进程；当原有的故障主进程恢复服务时，将被通知为备进程。主备进程值班状态切换，由各被管理进程自己负责。

目前系统管理中冗余管理进程需要管理的进程如下：

1. 实时库主备状态
2. 前置服务主备状态
3. HMI服务主备状态
4. 时序库主备状态
5. 配置服务主备状态
6. 通信中间件服务准备状态

其中各个模块的冗余管理机制，设计如下：

冗余管理模块主备裁决机制可以设计如下：

#### 2.6.5.1 系统管理冗余切换流程



系统管理冗余切换流程示意图

①：本机程序启动

②：向系统内其他节点广播主机登录消息

③：接收其他节点主服务应答

是：接收到主服务器应答，网内已有主服务器，本机转为备机状态，转向⑨

否：未收到主服务器应答，每间隔300ms进行重发登录信息，转向④

④：判断登录信息发送次数和主服务器应答，判断网内是否存在主服务器

是：重试次数大于2，网内不存在主服务器，本机成为主服务器，转向⑤

否：重试次数小于等于2，进行重发登录信息，转向②

⑤：本机成为主服务器（Master）。此状态收到其他主机登录消息时，立即回复本机Master状态的应答。转向⑥

⑥向网内广播平安报文，报告本机Master状态。转向⑦

⑦是否收到其他主机广播的Master平安报文，判断是否存在主机状态冲突

是：收到其他主机的Master平安报文，存在主机状态冲突。转向⑧

否：未收到其他主机的Master平安报文，当前系统各主机状态正常，转向⑤

⑧主机状态冲突，比较本地发送最后一个平安报文时报文中的时间戳和接收到的其他主机Master平安报文的时间戳。采用先判断出自己为主机、先发送平安报文的原则来决定Master的策略，报文中的时间戳精确到毫秒，可以认为时间戳再次冲突的概率极小，可以认为不会发生。

是：本机时间戳大于其他主机，根据竞争主备策略，本机状态转为备机（Slave），转向⑨

否：本机时间戳小于其他主机，根据竞争主备策略，本机仍然保留主（Master）状态，转向⑤

⑨本机状态为备（Slave），转向⑩

⑩广播本机Slave状态的平安报文，此报文用于网内其他机器判断本机是否离线。转向⑪

⑪接收其他主机的Master平安报文，并且对该报文进行计数，用于判断网内Master主机是否离线。

是：接收到Master平安报文，系统各主机准备状态正常，本机保持Slave状态，转向⑨

否：未接收到Master平安报文，统计未收到次数，转向⑫

⑫根据未收到Master平安报文次数，判断网内主服务器（Master）是否已经离线

是：计数小于等于2，在允许重试范围内，继续等待Master平安报文，转向⑨

否：计数大于2，判断网内主服务器（Master）已经离线，本机转为主（Master）状态，转向⑤。

#### 2.6.5.2 管理进程冗余切换流程

以实时库冗余切换为例，描述冗余切换流程。



实时库冗余切换流程示意图

### 主从数据同步

中心：集群间互备，集群内双机互备

车站：集群内双机互备

考虑到系统管理 对象数据相对简单，主备切换后 状态数据可以全部重新查询一次；

不再设计系统管理的同步服务；

### 系统配置

商用库+本地配置文件

尽量使用商用库进行系统配置。

### 安全管理

账号密码建立连接

### 日志记录

日志记录和归档

* 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
  2. 外部接口

与HMI存在查询交互接口

需要调用MQ相关API进行服务间消息调用；

此外冗余管理和进程管理 天然的需要和多个专业进程进行消息交互，此交互由系统管理定义接口，提供DLL动态库调用。

相关接口梳理如下

1. 进程管理相关接口（以下先使用C++开发当前定义，C调用方法一致）

* **进程注册接口**

int CProcMng::ProcRegister

(

const char \* proc\_name,

int start\_type = -1,

int interval = -1

)

**参数:**

[in] proc\_name 进程名或者服务名

[in] start\_type 启动类型（可不填）

• PROC\_R\_K 常驻关键进程

• PROC\_R\_UK 常驻非关键进程

• PROC\_UR\_K 非常驻关键进程

• PROC\_UR\_UK 非常驻非关键进程

• PROC\_ONCE 一次也不重启的进程

[in] interval 如果定时报告，报告周期（可不填）

**返回:**

1 注册成功

-1 注册失败

**注解:**

所有程序必须注册才会受到进程管理，不同的程序必须使用不同的进程名进行注册； 同一个程序如果需要启动多份，也必须使用不同的进程名，进程名的生成规则由注册方自定义。如果程序注册失败，应当退出。

当进程管理服务收到重复的进程名注册请求时，会判定当前注册请求失败。

1. **冗余管理相关接口（以下先使用C++开发当前定义，C调用方法一**致）

* **判断本机是否为指定进程的主机**

int CServicesManage::IsOnDuty (Const char \* proc\_name )

判断本机是否为指定进程的主机

**参数:**

[in] Const char \* proc\_name 进程名

**返回:**

* -1 执行错误
* 0 不是主机
* 1 是主机
* **获取本机上当前运行的进程名**

int CServicesManage::GetActiveProc ( vector<string> & proc\_name )

获取本机上活动的应用信息

**参数:**

[out] proc\_name 进程名结果的容器集合

**返回:**

活动的应用数量

**注解:**

* **获取某进程的主机信息或者第一个备机信息**

int CServicesManage::RequestService ( const char \* proc\_name

int policy,

char \* host\_name,

int domain\_id = -1

)获取本机上活动的应用信息

**参数:**

[in] proc\_name 需要查询的进程名

[in] policy 获取主机或备机

REQUEST\_DUTY 获取主机

REQUEST\_BACK 获取备机

[out] host\_name 获取的服务器名

[in] domain\_id 域号（如果policy为REQUEST\_BACK，domain\_id必须为-1）

**返回:**

* -1 执行错误
* 1 执行成功

**注解:**

如果domain\_id为-1时，本函数可以获取本域指定进程的主机或优先级最高的备机（如果存在）； 如果domain\_id为其他域号，本函数只能获取指定进程主机。

* **本机指定进程在运行**

如果该进程处理STOP或未配置该进程，返回false，否则返回true。

bool CServicesManage::isProcRun (const char\* proc\_name )

**参数:**

[in] proc\_name 进程名

**返回：**

判断结果

* **提供给进程管理使用的报告进程状态接口**

int CServicesManage:: reportPorcessInfo ( const char\* proc\_name，int proc\_state )

**报告进程状态**

**参数:**

[in] proc\_name 进程名

[in] proc\_state 进程状态

STATE\_NORMAL 0: 进程正常

STATE\_FAULT 1: 进程故障

**返回:**

1：报告成功

-1：报告失败

注解:

该接口的主要使用方应该是进程管理procmng

1. **节点信息相关接口（以下先使用C++开发当前定义，C调用方法一**致）

* **查询本机的domain\_id**

int CNodesManage::GetDefaultDomainID ( int & domain\_id )

**参数:**

[out] domain\_id 本机所属域

**返回:**

-1 执行错误

1 执行成功

1. 用户权限相关接口（以下先使用C++开发当前定义，C调用方法一致）

* **用户登录鉴权（输入用户名、密码进行鉴权）**

int CPermMng::SysLogin ( const string & user\_name,

const string & password,

const int userg\_id,

const int login\_sec

)

**参数:**

[in] const string & user\_name 用户名

[in] const string & password 用户口令

[in] const int userg\_id 用户组ID号

[in] const int login\_sec 用户登录时长，单位为秒

**返回:**

=0 成功

>0 失败，>0的返回值根据不同的情况定义不同的返回错误宏定义

<0 出错

* **用户注销（注销当前用户）**

int CPermMng::SysLogout ( void )

**参数:**

**[in] void**

**[out] int**

**返回:**

=0 成功

<0 失败 根据不同的失败情况，可以定义不同的错误宏定义

* **获取当前用户登录信息（用户ID驱动）**

int CPermMng::CurUser ( int & user\_id,

int & userg\_id,

int & level

)

参数:

[out] int & user\_id 当前登录用户的用户ID号

[out] int & userg\_id 当前登录用户所属用户组ID号

[out] int & level 当前登录用户的用户级别

返回:

=0 成功

PERM\_NO\_USER\_LOGIN 无用户登录

<0 出错

* **获取当前用户登录信息（用户名驱动）**

int CPermMng::CurUser ( string & user\_name,

string & userg\_name,

int & level

参数:

[out] string & user\_name 当前登录用户的用户名

[out] string & userg\_name 当前登录用户所属用户组名

[out] int & level 当前登录用户的用户级别

返回:

=0 成功

PERM\_NO\_USER\_LOGIN 无用户登录

<0 出错

* **判断指定的用户是否具有某一权限**

int CPermMng::PermCheck (const &string user\_name, const int perm\_type, char \* arg ）

单条记录的权限判断。具体传入的权限类型待定，目前初步定为普通权限和扩展权限（扩展权限包括报警确认、遥控操作等）；

参数:

[in] const &string user\_name 查询的用户名

[in] const int perm\_type 权限类型

[in] char \* arg 权限类型对应值的指针，如 int dept\_id = 1, char \*arg = (char \*)&dept\_id

返回:

PERM\_PERMIT 具有权限

PERM\_FORBIT 没有权限

PERM\_NO\_USER\_LOGIN 无用户登录

<0 出错

* **判断指定的用户是否具有某组权限**

当前设计的判断逻辑中，某用户的可登录位置和权限位置相对应，权限不具备位置属性。

int CPermMng::PermCheck ( const & string user\_name,

const int perm\_type,

vector< char \* > & arg,

vector< int > & ret

)

多条记录的权限判断，所有记录必须属于同一种权限类型

参数:

[in] const & string user\_name 指定的查询用户名

[in] const int perm\_type 权限类型（画面类型、报表类型等）

[in] vector <char \* > & arg 多条记录对应值的指针

[out] vector<int> & ret 针对每条记录值的权限判断结果，长度与记录数相同。

成功则为权限信息：具有权限--PERM\_PERMIT，没有权限--PERM\_FORBIT，出错<0

返回:

PERM\_PERMIT 具有权限

PERM\_FORBIT 没有权限

PERM\_NO\_USER\_LOGIN 无用户登录

<0 出错

* **判断指定的用户是否具有某组权限**

当前设计的判断逻辑中，某用户的可登录位置和权限位置相对应，权限不具备位置属性。

int CPermMng::PermCheck ( const & string user\_name,

const int perm\_type,

vector< char \* > & arg,

vector< int > & ret

)

多条记录的权限判断，所有记录必须属于同一种权限类型

参数:

[in] const & string user\_name 指定的查询用户名

[in] const int perm\_type 权限类型（画面类型、报表类型等）

[in] vector <char \* > & arg 多条记录对应值的指针

[out] vector<int> & ret 针对每条记录值的权限判断结果，长度与记录数相同。

成功则为权限信息：具有权限--PERM\_PERMIT，没有权限--PERM\_FORBIT，出错<0

返回:

PERM\_PERMIT 具有权限

PERM\_FORBIT 没有权限

PERM\_NO\_USER\_LOGIN 无用户登录

<0 出错

* 获取当前登录用户的特殊功能责任区和车站信息

int CPermMng::GetSpeFunc ( const string & spe\_func\_def,

vector< int > & region\_id,

vector< int > & station\_id

)

参数:

[in] const string & spe\_func\_def 特殊功能宏定义

[out] vector <int> & region\_id 责任区ID

[out] vector <int> & station\_id 车站ID

返回:

=0 成功

PERM\_NO\_USER\_LOGIN 无用户登录

<0 出错

* 锁定用户，日志号90201~90205

int CPermMng::LockUser ( const int user\_id )

参数:

[in] int user\_id 用户ID号

返回:

=0 成功

<0 出错

* 判断用户登录当前是否已超时

int CPermMng::IsCurUserTimeout ( void ) 参数:

参数:

[in] void

返回:

=0 未超时

PERM\_NO\_USER\_LOGIN 无用户登录

PERM\_USER\_TIMEOUT 超时

PERM\_USER\_TIMEOUT\_REMIND 超时提醒

<0 出错

int CPermMng::CheckPassword ( const string & password )

* 验证当前登录用户密码，日志号90231~90234

参数:

[in] const string & password 输入密码

返回:

<0 出错

PERM\_FORBIT 密码错误

PERM\_PERMIT 密码正确

PERM\_NO\_USER\_LOGIN 无用户登录

) 3.3 内部接口

系统管理自身使用的内部接口，主要有如下几个：

1、根据PID查询实例名

int CProcMng::getInstanceName ( pid\_t pid, char \* pinstance\_name )

通过进程ID获取进程实例名

参数:

[in] pid 进程号

[out] pinstance\_name 实例名

返回:

* 1 成功
* -1 失败

进程管理服务在接收到查询请求后，调用相关的操作系统接口，获取实例名；

2、根据实例名查询进程编号

int CProcMng::getInstancePid ( char \* pinstance\_name , ) [in] pinstance\_name 实例名

[out] 进程ID

返回:

* 1 成功
* -1 失败

进程管理服务在接收到查询请求后，调用相关的操作系统接口，获取进程ID编号；

# 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。