系统管理平台

**系统设计说明书**

SMP\_SSDD\_V0.1

拟 制

审 核

标准化

批 准

深圳市中航比特通讯技术有限公司

**修订历史记录**

**A** - 增加 **M** - 修订 **D** - 删除

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变更  版本号 | 日期 | 变更  类型 | 修改人 | 摘要 | 备注 |
| 0.1 | 2013.04.20 | A | 胡新宇 | 创建文档 |  |
| 0.2 | 2013.6.15 | M | 胡新宇 | 修改文档 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目 录

[1 范围 1](#_Toc362616062)

[1.1 标识 1](#_Toc362616063)

[1.2 系统概述 1](#_Toc362616064)

[1.3 文档概述 2](#_Toc362616065)

[2 引用文档 2](#_Toc362616066)

[3 系统级设计决策 2](#_Toc362616067)

[3.1 系统管理平台根据产品部署情况划分管理范围 2](#_Toc362616068)

[3.2 系统管理平台分为系统管理服务器与单板系统管理模块两个层级 4](#_Toc362616069)

[3.3 轻量级的系统管理服务器，提供业务拓展的支持 7](#_Toc362616070)

[3.4 管理范围内允许部署多个系统管理服务器，且被被看做一个逻辑整体 8](#_Toc362616071)

[3.5 系统管理平台的管理通讯网络划分为三个子网 9](#_Toc362616072)

[3.6 系统管理平台跨管理范围时信令传递采用转发方式 10](#_Toc362616073)

[3.7 系统管理平台信令目标对象进行编码并保存转发信息，作为信令传递依据 11](#_Toc362616074)

[3.8 系统管理平台采用UDP通信方式 12](#_Toc362616075)

[3.9 系统管理平台中，配置数据交互采用XML方式组织 12](#_Toc362616076)

[4 系统体系结构设计 12](#_Toc362616077)

[4.1 系统部件 13](#_Toc362616078)

[4.1.1 系统管理服务器 13](#_Toc362616079)

[4.1.2 单板系统管理模块 18](#_Toc362616080)

[4.2 数据模型 22](#_Toc362616081)

[4.2.1 数据模型 22](#_Toc362616082)

[4.2.2 通信模型 35](#_Toc362616083)

[4.3 执行方案 45](#_Toc362616084)

[4.3.1 数据配置 45](#_Toc362616085)

[4.3.2 故障管理 53](#_Toc362616086)

[4.3.3 业务管理 61](#_Toc362616087)

[4.3.4 系统监控 62](#_Toc362616088)

[4.3.5 信息上报 65](#_Toc362616089)

[4.3.6 软件管理 69](#_Toc362616090)

[4.3.7 系统安全 83](#_Toc362616091)

[4.3.8 数据同步 85](#_Toc362616092)

[4.3.9 信令传递与转发 88](#_Toc362616093)

[4.4 接口设计 93](#_Toc362616094)

[4.4.1 接口标识和图表 93](#_Toc362616095)

[4.4.2 外部接口描述 94](#_Toc362616096)

[4.4.3 内部接口描述 94](#_Toc362616097)

[5 需求的可追踪性 94](#_Toc362616098)

[5.1 部署需求 94](#_Toc362616099)

[5.2 数据配置功能需求 95](#_Toc362616100)

[5.3 故障管理功能需求 95](#_Toc362616101)

[5.4 业务管理功能需求 95](#_Toc362616102)

[5.5 系统监控功能需求 96](#_Toc362616103)

[5.6 系统信息上报功能需求 96](#_Toc362616104)

[5.7 软件管理功能需求 96](#_Toc362616105)

[5.8 系统安全功能需求 97](#_Toc362616106)

[6 注释 97](#_Toc362616107)

# 范围

## 标识

SMP： System Manage Platfram 系统管理平台

版本： V0.1

适用时： 软件设计、软件实现、单元测试、集成测试、系统测试、合格性测试

## 系统概述

系统管理平台系公司独立项目，为模块化通信平台的组成部分，主要为模块化通讯平台相关产品提供管理与维护功能，以方便用户对产品进行相关的管理与维护操作。

系统管理平台的目标是为模块化通信平台产品提供通用化的系统管理功能支撑，对产品中系统管理的功能需求进行抽象化，使用通用的逻辑进行处理，保持系统管理业务逻辑上的一致，以屏蔽不同产品在数据上的差异性。要做到在产品的具体实施过程中，只需要对系统管理平台做一些必要的配置，即可实现并满足产品对系统管理方面的功能需求。

系统管理平台需要有支持灵活部署的能力，以适配不同产品的不同部署需求。要能够适应简化的部署方式，也要能够适应多层级的部署方式。

系统管理平台对外提供统一功能接口，包括系统管理对客户端接口以及对业务应用接口。

系统管理平台主要包括以下功能：数据配置功能、故障管理功能、业务管理功能、系统监控功能、系统信息上报功能、软件管理功能、系统安全功能等。

图1‑1 系统管理平台在产品中的位置

## 文档概述

本文档用于指导系统管理平台的软件设计，也可供系统管理平台系统测试人员系统测试进行参考。

# 引用文档

SMP\_SSS\_V1.0 《系统管理平台—系统规格说明书》

# 系统级设计决策

## 系统管理平台根据产品部署情况划分管理范围

我们将一个产品范围内，所有需要被管理的实体，例如业务软件、硬件设备、交换网络等，称之为**管理对象**。

由于产品架构是可变的，每个产品的部署情况都有可能不同，面对不同产品的部署方式的不同，以及管理对象的不同，如何对其提供清晰明了的管理服务，这是系统管理平台需要着重考虑的重点。

要提供清晰明了的管理服务，首先就是要明确管理关系，知道某个管理对象应该被谁管理。

从管理对象的角度来看，无论一个产品以何种方式部署，总是可以将管理对象以其放置的位置、运行关系，划分为一个或若干个集合，集合之间是相互独立的，不存在交集。例如下图：



图3‑1 产品管理范围划分示意图

我们可以将这些由位置、关系划分的，不同的由管理对象构成的，相互之间独立的集合称之为**管理范围**。将产品划分为一个或若干个管理范围，每一个管理范围覆盖一部分管理对象。

从管理范围的角度来看，每一个管理范围都应该具备独立运行的能力，以及对自身进行管理的自治能力，而不依赖于其他管理范围。

从管理对象的角度而言，一个管理对象只归属于一个管理范围，和其他管理范围不存在管理关系。该管理对象只接受于本管理范围提供的管理服务，只要本管理范围提供相应的系统管理服务，即可正常运行。

划分管理范围的优点是可以明晰组织结构，避免管理上的重叠，提高管理范围之间的独立性，实现管理范围自治、避免相互之间的关系与依赖。

需要说明的是：

1、产品的管理范围划分，是根据产品的实际部署情况确定的。当确定了管理范围的划分，应修改相应的配置信息，系统管理平台通过读取配置信息了解相应的管理范围划分，分别实施管理。

2、在产品管理范围的划分上，应当注意就近原则，尽量将运行在一起的硬件设备划分到一个管理范围内，以避免由于运行不同步，而导致系统管理平台的服务中断。

## 系统管理平台分为系统管理服务器与单板系统管理模块两个层级

通信产品的系统管理体系，按照从高到低的顺序，大致可分为三个层级，分别为：用户界面层、平台管理层、管理对象层。

对于每一个管理范围而言，我们在平台管理层可以部署一个管理单元，其功能是为管理范围内的所有管理对象提供系统管理服务，其所用域仅限于所处的管理范围。对于该管理单元，我们将其命名为**系统管理服务器**。

一个管理范围覆盖若干个管理对象，而这些管理对象可能运行在不同的硬件设备上。这样的话，如果仅有系统管理服务器，无法做到跨硬件设备、跨操作系统进行管理的目的，因此我们还需要在平台管理层部署一类管理单元，该管理单元为所处的硬件设备上运行的所有管理对象提供系统管理服务，其作用域为与其同一操作系统下的所有管理对象。对于该管理单元，我们将其命名为**单板系统管理模块**。

这样的话，从层次上来看，我们可以将平台管理层分为两个子层，分别是服务器子层和管理模块子层，系统管理平台跨两个子层，在服务器子层对应的是系统管理服务器，在管理模块子层上对应的是单板系统管理模块。如下图所示：



图3‑2 系统管理平台层次划分示意图

系统管理服务器位于服务器子层，从层次上来看，与各个层次关系如下：



图3‑3 系统管理服务器与各个层次关系示意图

系统管理服务器与用户界面层之间的关系，主要是与客户端进行交互，提供以下功能:

1. 系统管理服务器为客户端提供配置数据源，并提供配置数据的获取、更新、数据回滚、数据备份等功能。
2. 系统管理服务器为客户端上报故障信息，并提供故障信息查询及维护功能。
3. 系统管理服务器为客户端上报事件信息，并提供事件信息查询及维护功能。
4. 系统管理服务器为客户端提供业务管理信令的传递通道。
5. 系统管理服务器保证客户端的接入、访问与操作的安全性和合法性。
6. 系统管理服务器负责与客户端之间的信令传输功能。

系统管理服务器与管理模块子层之间的关系，主要是与单板系统管理模块进行交互，并提供以下功能:

1. 系统管理服务器作为配置数据源，保存本管理范围内的所有管理对象的配置数据。对单板系统管理模块提供配置数据的获取、校验、更新等功能。
2. 系统管理服务器接收单板系统管理模块上报的故障信息，并对其进行保存。
3. 系统管理服务器接收单板系统管理模块上报的事件信息，并对其进行保存。
4. 系统管理服务器与单板系统管理模块之间保证业务管理信令的传输通道，保证业务管理信令的正确传递。
5. 系统管理服务器与单板系统管理模块间保证归属关系的合法性。
6. 系统管理服务器与单板系统管理模块间的信令传输功能。

系统管理服务器与同处服务器子层的其他系统管理服务器间，主要是服务器间的数据同步，其功能如下：

1. 系统管理服务器间，根据服务器间关系，进行相关数据的同步处理。
2. 对于跨管理范围的系统管理指令，系统管理服务器间进行信令的路由转发处理。

单板系统管理模块位于模块管理子层，从层次上来看，与各个层次关系如下图：



图3‑4 单板系统管理模块与各个层次关系示意图

单板系统管理模块与服务器子层之间的关系，主要是与所属的系统管理服务器进行交互，提供以下功能：

1. 向系统管理服务器发起配置数据校验，与系统管理服务器进行配置数据同步处理，接收系统管理服务器下发的配置数据更新。
2. 故障信息上报系统管理服务器。
3. 事件信息上报系统管理服务器。
4. 与系统管理服务器之间进行业务管理信令的传递。
5. 与系统管理服务器之间的信令传输。

单板系统管理模块与管理对象层之间的关系，主要是与本单板上运行的管理对象进行交互，并提供以下功能：

1. 对管理对象进行配置数据更新通知。
2. 提供故障上报接口、接收管理对象上报的故障信息。
3. 提供事件上报接口、接收管理对象上报的事件信息。
4. 与管理对象进行业务管理信令的传递与交互。
5. 对管理对象软件进行管理功能，包括软件启动、运行、退出的管理，以及软件更新处理。

在同一管理范围内，系统管理服务器与单板系统管理模块部署关系，如图下图所示：



图3‑5管理范围内系统管理服务器与单板系统管理模块关系示意图

## 轻量级的系统管理服务器，提供业务拓展的支持

由于系统管理服务器可能被部署在产品的某单板上运行，考虑到单板在处理能力和存储容量上无法与专业的服务器相比。在进行某些较为复杂的处理时，可能会对运行环境造成较大的冲击。

因此在系统管理服务器设计上，应注重轻量化处理，系统管理服务器仅提供必要的系统管理服务，以及相应的数据存储。能够对管理范围内的管理对象的正常运行提供必要的支撑，以及对其进行必要的管理服务。

对于一些非必须的管理业务，可以在考虑在系统管理平台之外，增加相应的服务软件实体来实现，例如故障服务器、日志服务器等。这类软件实体对于系统管理平台来说，可以以客户端的方式，接入并访问系统管理平台，从而实现其相应的功能。

对于这类软件实体，系统管理服务器本身提供必要的数据支持，或处理通道。例如：对于故障服务器，系统管理服务器对其提供故障信息数据，由系统管理服务器主动将故障信息上报给故障服务器，也支持故障服务器向系统管理服务器查询故障信息。

系统管理服务器与其他管理软件实体关系如下图所示：



图3‑6 系统管理平台支持其他管理软件实体接入示意图

## 管理范围内允许部署多个系统管理服务器，且被被看做一个逻辑整体

考虑到系统管理服务器负责一个管理范围内的所有系统管理服务，一旦该系统管理服务器运行异常，则该管理范围内的所有系统管理服务将中断。

为了避免由于系统管理服务器运行异常导致整个管理范围的系统管理服务中断，可以考虑在一个管理范围内，部署两个或以上的系统管理服务器。这样，即使一个系统管理服务器故障后，仍然有系统管理服务器为本管理范围提供系统管理服务。

从逻辑角度来看，无论当前是那个系统管理服务器提供服务，从系统管理服务本身来看是相同的，客户端和单板系统管理模块连接到任一系统管理服务器，都可以接收到相同的系统管理服务。因此在逻辑上，我们可以将多个系统管理服务器看做是一个提供系统管理服务的逻辑整体。



图3‑7 管理范围内多系统管理服务器组成一个逻辑整体

从物理角度来看，同一管理范围中的每个系统管理服务器，都有各自的通讯链路，客户端和单板系统管理模块通过其中任意一个通讯链路与相应的系统管理服务器进行通讯交互。

结合两方面，我们可以将部署于同一管理范围内的多个系统管理服务器看做一个逻辑服务器，该逻辑服务器对外有多条通讯链路，客户端和单板系统管理模块可以通过任一链路，与逻辑系统管理服务器进行通讯交互。

## 系统管理平台的管理通讯网络划分为三个子网

系统管理平台在通讯方面，根据其通讯关系，可划分为以下三类通讯：系统管理服务器与客户端通讯、系统管理服务器与单板系统管理模块通讯、系统管理服务器间通信。

由上述三种通讯关系，我们可以将系统管理平台的通讯网络划分为三个子网。



图3‑8 系统管理平台通讯子网划分

如上图所示：系统管理平台的通讯网络被划分为3个部分，分别是客户端访问通讯子网、管理范围间通信子网、管理范围内通讯子网。

客户端访问通讯子网主要负责客户端与系统管理服务器间的网络通信。在该子网中，必须为客户端与系统管理服务器分配通讯地址，并保证二者之间的通讯畅通。

管理范围间通信子网主要负责系统管理服务器之间的网络通讯。如果产品部署了多个系统管理服务器，则在本子网中，必须保证中所有的系统管理服务器相互之间通讯畅通。

管理范围内通讯子网主要负责一个管理范围内，系统管理服务器与单板系统管理模块间的网络通讯。在本子网中，必须保证该管理范围内，系统管理服务器与所有的单板系统管理模块之间通讯的畅通。

注：在具体产品中，根据产品需求和部署情况，上述三个子网相互之间在物理上存在融合和合并的可能。

## 系统管理平台跨管理范围时信令传递采用转发方式

由于从用户角度看，系统管理平台为一个整体，因此系统管理平台需要做到用户接入到系统管理任一接入点，都能够给对产品范围内的任一管理对象进行管理。

当产品划分为多个管理范围时，当指令接收的系统管理服务器与管理目标不处于同一管理范围时，就产生了跨管理范围通讯的问题。

如之前章节描述，每个管理范围都是独立而自治的，每个管理范围内的单板系统管理模块只与本管理范围内的系统管理服务器进行交互。基于上述原则，处于管理模块子层的单板系统管理模块不涉及跨管理范围通讯的处理。因此我们在服务器子层处理跨管理范围通讯的问题。

当发生跨管理范围通讯时，接收到指令的系统管理服务器将指令转发给目标管理范围的系统管理服务器，再由其将指令下发给目标单板系统管理模块进行处理。如下图所示：



图3‑9 跨管理范围的信令传递

如上图所示：客户端接入管理范围1的系统管理服务器1，当下发对业务应用B的管理指令时，客户端将指令发送至系统管理服务器1，信令由系统管理服务器1路由转发至业务应用B所在的管理范围2的系统管理服务器2，再由系统管理服务器2进行信令下发。

需要强调的是：信令转发行为只存在于系统管理服务器之间，即局限于服务器间通信子网。

## 系统管理平台信令目标对象进行编码并保存转发信息，作为信令传递依据

在系统管理信令包的传递过程中，决定信令传递走向的关键信息为发送源与发送目的，因为这决定了信令由谁发送，要发送到哪去。

而在系统管理体系中，信令传递经过的每一个环节，对于对信令发送源与发送目的信息的理解应该是统一的，否则将无法对信令作出正确的传递。因此，有必要在系统管理体系中，对信令的发送源与发送目的按照统一规则进行编码。

系统管理平台根据产品部署情况，对系统管理服务器、单板系统管理模块以及各种管理对象进行统一的分类编码。编码定义要求每一个信令的传递环节都能够正确识别。

编码方式是根据产品具体实施确定的，可以通过配置的方式被系统管理平台读取并掌握。

信令路由信息由相应的产品系统信息以及客户端登录信息构成。

系统信息以配置数据形式存在，被系统管理平台读取并理解。系统配置信息主要包含产品管理范围的划分、管理范围内系统管理服务器的部署情况，管理对象的归属等信息。系统信息主要作为下行信令的传递依据。

客户端登录信息是动态的，当有客户端登录系统管理服务器及产生相应的登录信息，并同步给其他系统管理服务器。客户端登录信息主要作为上行信令的传递依据。

信令转发信息在系统管理平台内的所有系统管理服务器间共享，由于客户端登录情况会发生变化，转发信息是实时变化的。当转发信息发生变化时，需要同步到每一个系统管理服务器。

## 系统管理平台采用UDP通信方式

由于系统管理平台与相关业务应用可能不运行在同一设备同一操作系统，甚至系统管理平台自身的各个组成部分都可能不运行在同一设备上或同一进程中，因此系统管理平台在通讯方式应使用进程间通讯方式。

基于TCP/IP协议的UDP协议，只需要系统管理平台中的每一个系统管理实体具备TCP/IP协议栈，有相应的IP地址和端口分配，即可保证通信，且相关技术成熟，成本小，能够满足系统管理平台分步式部署。

UDP通信的缺点是无法保证传输的可靠性，这一点可以通过增加额外的保障手段，来提高通信可靠性。

基于实现的难易程度及利弊分析，系统管理平台通讯可以采用UDP方式通讯。

## 系统管理平台中，配置数据交互采用XML方式组织

通讯产品的各种配置数据需求变化性很大，有各种组织形式的数据配置需求。在这种情况下，系统管理平台需要能够迅速支持各种数据配置需求。

XML作为可扩展的标记性语言，具备可读性好、扩展性强、内容与结构分离、适合传输与存储等特点，已被广泛使用于配置数据的存储。

基于XML的拓展性强、适合传输与存储的特点，系统管理平台可以采用XML的方式组织配置数据在平台内部的存储与传输，以及对外交互。

这样系统管理平台只需实现XML数据解析及相关处理，这样无论配置需求如何变化，系统管理平台依然可以适应，并进行高效的处理。

# 系统体系结构设计

系统管理平台软件架构图，如图4-1所示：



图4‑1系统管理平台软件架构图

系统管理平台分为系统管理服务器与单板系统管理模块两个组成部件。

系统管理服务器负责对上与客户端交互，接收用户下发的指令，将指令处理结果反馈给用户，系统管理服务器对下为所在管理范围提供系统管理服务，例如配置管理、故障管理、安全管理等。

单板系统管理模块与业务应用交互，为所在单板的业务应用提供相关系统管理服务，例如软件管理、数据配置、故障上报等。

## 系统部件

### 系统管理服务器

系统管理服务器需要实现数据同步功能、数据配置功能、故障管理功能、业务管理功能、信息上报功能、安全管理功能、通信及信令传递功能。图4-2中，彩色部分标识了系统管理服务器的功能结构。



图4‑2系统管理服务器功能结构图

#### 数据配置功能

系统管理服务器作为配置数据源，为系统管理客户端和单板系统管理模块提供配置数据源服务，主要功能如下：

1. 支持配置数据存储功能。
2. 支持客户端查询配置数据。
3. 支持客户端下发配置数据更改操作，包括配置数据添加、修改及删除。
4. 支持单板系统管理模块获取配置数据。
5. 支持与单板系统管理模块进行配置数据校验，保持配置数据一致性。
6. 支持配置数据备份与恢复功能。

系统管理服务器数据配置功能设计点如下：

1. 配置数据以XML文件形式保存，此为系统管理服务器数据配置功能实现的核心。
2. 系统管理服务器按照业务应用的类型和标识组合的规则，指定配置文件的保存路径。
3. 系统管理服务器按照业务应用的类型、标识、及配置表ID组合的规则，指定配置数据XML文件的名称。
4. 系统管理服务器通过XML解析库，解析XML配置文件内容。
5. 系统管理服务器启动时需读取各个配置文件内容，并将其保存在本地内存中。
6. 当收到配置数据获取请求时，系统管理服务器从内存中取出相关数据，返回给对方。
7. 当收到配置数据更新指令时，导致配置数据发生改变时，系统管理服务器更新内存中相关配置数据，并重新生成配置文件。
8. 系统管理服务器与单板系统管理模块之间，以文件形式交互配置数据，当配置数据变化时，系统管理服务器将相应的配置文件内容通过系统管理信令下发至单板系统管理模块，单板系统管理模块收到后，更新本地配置文件。
9. 系统管理服务器与单板系统管理模块之间的配置数据校验，采用比对相关配置文件的MD5码实现。

#### 故障管理功能

系统管理服务器提供故障上报的通道，处理单板系统管理模块上报的故障信息，支持系统管理客户端对故障信息进行查询，并提供故障信息的相关维护功能，主要功能如下：

1. 支持单板系统管理模块上报故障信息的处理。
2. 支持一定数量的故障信息存储。
3. 支持客户端检索故障信息。
4. 支持客户端对故障信息进行备份。

系统管理服务器故障管理设计点如下：

1. 系统管理服务器将故障信息记录分为两种，分别是当前故障信息和历史故障信息。所有的故障信息记录保存在系统管理服务器本地数据库中。
2. 当系统管理服务器收到故障信息时将其保存为当前故障信息，当收到相应的故障恢复信息时，将其转存为历史故障信息。
3. 系统管理服务器根据故障ID、故障类型、故障对象类型、故障对象ID来作为故障信息匹配的依据。
4. 系统管理服务器根据上述故障匹配依据，来匹配故障恢复信息对应的故障信息，并过滤重复故障信息。
5. 当系统管理服务器收到非本管理范围对象上报的故障信息时，仅将其发送至连接到本地的客户端，并不做存储处理。
6. 系统管理服务器对故障信息的检索操作，采用操作数据库记录的方式实现。
7. 系统管理服务器保存一定数量的历史故障信息，当历史故障信息记录超过规定数量后，系统管理服务器将清除这些历史故障记录。

#### 业务管理通道

系统管理服务器提供业务管理信令的传输通道，主要功能为保证业务管理信令的可靠传递。业务管理信令传输通道，由通讯及信令传递模块实现其功能，并保证通道的可靠性。

#### 系统监控功能

系统管理服务器提供系统监控信令的传输通道，主要功能为保证业务管理信令的可靠传递。

系统管理服务服务器监控信令传输通道，由通讯及信令传递模块实现其功能，并保证通道的可靠性。

系统监控的门限值，及监控对象的配置，以配置数据的处理机制实现。

系统监控结果的查询及响应，可以通过业务管理通道进行传递。

#### 信息上报功能

系统管理服务器为管理范围内的业务应用提供信息上报功能，将各个业务应用产生的事件信息上报到系统管理客户端，并提供上报信息的保存和维护功能，主要功能如下：

1. 提供事件信息上报通道，支持将单板系统管理模块上报的事件信息上报到客户端。
2. 提供事件信息的保存机制，允许保存一定数量的事件信息。
3. 支持客户端对事件信息的检索功能。
4. 支持事件信息的备份功能。

系统管理服务器事件管理设计点如下：

1. 系统管理服务器将上报事件信息记录保存在系统管理服务器本地数据库中。
2. 当系统管理服务器收到非本管理范围对象上报的事件信息时，仅将其发送至连接到本地的客户端，并不做存储处理。
3. 系统管理服务器对事件信息的检索操作，采用操作数据库记录的方式实现。
4. 系统管理服务器保存一定数量的事件信息，当事件信息记录超过保存数量后，系统管理服务器将清除这些事件记录。

#### 系统安全功能

系统管理服务器负责系统对外的安全认证功能，对收到的客户端登录请求进行权鉴处理，告知其相关权限。主要功能如下：

1. 提供客户端登录权鉴机制，负责对客户端发送的登录请求进行权鉴操作。
2. 权鉴机制是具有持续性的，对于通过权鉴后的用户下发的指令，依然要对其进行权鉴检查，查看其是否具有相应的权限。
3. 提供用户权限配置功能，支持用户对权限信息进行修改。
4. 用户权限告知，当用户信息通过权鉴后，系统管理服务器将告知用户其所拥有的操作权限。
5. 安全信令功能，确保只处理通过权鉴的用户下发的指令被处理。
6. 安全配置功能，支持用户修改安全相关配置。

系统管理服务器事件管理设计点如下：

1. 系统管理服务器以配置数据方式，处理系统安全相关配置数据。
2. 系统管理服务器制定一个特定规则，为权鉴成功的用户生成通行码，该通行码在权鉴响应消息中，返回给客户端。该通行码应该能够被其他系统管理服务器进行校验，判断其合法性。
3. 系统管理服务器支持用户获取其权限信息，当用户发送权限信息请求时，系统管理服务器将其拥有的权限信息返回给用户。

#### 数据同步功能

系统管理服务器具备数据同步功能，负责与其他系统管理服务器进行各种业务数据的同步操作，以保证各种业务数据的一致性。主要功能如下：

1. 软件管理相关配置数据同步。
2. 软件业务配置数据同步。
3. 安全信息数据同步。
4. 产品系统信息数据同步。
5. 客户端登录信息同步。

#### 通信及信令传递转发功能

系统管理服务器具备信息通信以及信息传递转发功能，负责保证各种系统管理信令在系统管理平台中的正确传送，将其正确的送至其目的地。主要功能如下：

1. 通信链路的建立与维护。
2. 系统管理信令的接收与发送。
3. 信令接收缓冲功能。
4. 转发信息的保存与维护。
5. 通信链路健康性检测以及对转发信息的影响。
6. 信令发送方式的选择。
7. 系统信息的在系统管理服务器间的共享。
8. 客户端登录信息在系统管理服务器间的共享。

系统管理服务器通信及信令转发设计点如下：

1. 系统管理服务器读取产品系统配置信息，了解产品构成、管理范围的划分以及各个管理范围内的系统管理服务器的通信地址。
2. 系统管理服务器之间，共享客户端登录信息。
3. 以上两种信息被称为转发信息，用于系统管理服务器作为信令转发的依据，当系统管理服务器收到系统管理信令时，根据转发信息决定该信令的发送方式（传递或转发）及发送目的（客户端、单板系统管理模块、其他系统管理服务器）。
4. 当客户端登录或登出系统管理服务器时，系统管理服务器之间及时同步客户端登录信息，以便共享客户端登录信息。
5. 系统管理服务器在信令传递方面，采用单向滑窗机制，以保证信令传输的效率。

### 单板系统管理模块

单板系统管理模块需要实现数据配置功能、、系统监控功能，软件管理功能以及故障上报通道、业务管理通道、信息上报通道。图4-3中，彩色部分标识了单板系统管模块的功能结构。



图4‑3单板系统管理模块功能结构图

#### 数据配置功能

单板系统管理模块为所在单板上的业务应用软件提供数据配置功能服务，主要功能如下：

1. 支持通过XML文件为业务应用提供配置数据。
2. 支持XML配置文件的生成和更新功能。
3. 支持配置更新通知功能，当配置数据发生改变时，通知业务应用。
4. 支持与系统管理服务器进行配置数据校验，保持配置数据一致性。

单板系统管理模块数据配置功能设计点如下：

1. 配置数据以XML文件形式保存在单板系统管理模块指定路径下。
2. 单板系统管理模块对配置文件的保存路径及XML文件名称的规则，与系统管理服务器一致。
3. 单板系统管理模块和业务软件之间的配置数据交互以XML文件方式。
4. 当配置数据因用户发送指令发生改变时，单板系统管理模块根据下发的配置数据内容，更新配置文件，并通知业务应用配置数据变更。
5. 配置数据校验请求由单板系统管理模块向有系统管理服务器发起，通过比对相关配置文件的MD5码实现。

#### 系统监控功能

单板系统管理模块为所在单板提供资源监控功能，主要功能如下：

1. 支持对所在单板资源使用情况的采集功能，定时获取单板的CPU与内存占用率数值。
2. 支持对所在单板资源使用情况的监控功能，当CPU与内存的占用率达到门限值时，进行故障信息上报，对用户进行通知。
3. 支持对门限值配置功能，处理系统管理服务器下发的门限值修改指令，用新的门限值取代默认的门限值。
4. 支持监控结果查询功能，处理系统管理服务器下发的监控结果查询指令，将当前的单板CPU与内存占用率返回给用户。

单板系统管理模块系统监控设计点如下：

1. 单板系统管理模块通过操作系统接口，获取系统资源监控信息。
2. 单板系统管理模块以配置数据的处理方式，处理系统监控门限值配置数据。
3. 当每周期的系统资源数据采集到后，单板系统管理模块将数据与门限值进行比对，一旦超出，则以故障信息的方式上报用户。
4. 当用户下发系统资源数据获取请求后，单板系统管理模块将数据返回给用户。

#### 软件管理功能

单板系统管理模块为所在单板的业务应用提供软件管理功能，为业务应用软件提供启动、软件运行监控、软件退出、软件更新、软件运行控制等功能，主要功能如下：

1. 支持业务应用软件及系统管理平台自身软件的管理。
2. 具备系统管理平台软件自身运行故障恢复机制。
3. 具备软件管理配置信息获取，获取本地软件管理所需的相关信息。
4. 具备软件启动管理功能，能够对业务应用软件启动过程进行监控，当软件启动失败时，按照设定的软件管理策略对其进行处理。
5. 具备软件运行监控功能，对运行中的业务应用进行监控，检查业务软件运行是否正常，及业务软件运行异常时，根据相应的软件管理策略进行处理。
6. 具备软件退出功能，能够停止业务应用软件的运行。
7. 具备软件运行控制功能，允许用户控制业务应用软件的运行状态，人为的启动和停止业务软件的运行。
8. 具备软件管理策略的配置功能，允许用户对相关的软件管理策略进行配置。
9. 具备文件上传下载功能，文件下载功能为软件更新使用，文件上传功能为软件导出使用。
10. 具备业务应用软件更新功能，允许用户对业务应用软件进行软件更新操作，通过文件下载功能，将新的业务应用软件文件下载到单板上，更新老的文件。
11. 支持软件导出功能，允许用户对单板上的业务应用软件进行导出备份，将业务应用软件文件通过文件上传功能，将文件导出，备份到用户侧。

单板系统管理模块软件管理设计点如下：

1. 软件管理策略信息以配置数据的形式进行读取、保存及维护。
2. 单板系统管理模块启动时即读取软件管理策略信息，作为软件管理处理的依据。
3. 单板系统管理模块调用操作系统接口，启动业务软件进程，来实现软件启动控制。
4. 单板系统管理模块通过通知业务软件退出，或调用操作系统接口杀死业务软件进程，来实现软件退出控制。
5. 软件更新过程分为以下几个步骤：软件验证、软件更新、软件启动、软件回滚；其中软件验证和软件回滚步骤由客户端负责，系统管理平台负责软件更新和软件启动步骤。
6. 当软件更新后，启动失败，单板系统管理模块需阻止该软件继续启动，并通过故障信息通知用户，软件更新失败。由用户介入，进行软件回滚。

#### 故障上报通道

单板系统管理模块提供故障信息上报传输通道，主要保证故障信息的可靠传递。故障信息传输通道，由通讯模块实现其功能，并保证其可靠性。

#### 事件上报通道

单板系统管理模块提供事件信息上报传输通道，主要保证事件信息的可靠传递。事件信息上报通道，由通讯模块实现其功能，并保证其可靠性。

#### 业务管理通道

单板系统管理模块提供业务管理信令的传输通道，主要功能为保证业务管理信令的可靠传递。业务管理信令传输通道，由通讯模块实现其功能，并保证其可靠性。

#### 通信功能

单板系统管理模块需具备信息通信功能，负责保证各种系统管理信令的正确传送，将其正确的送至其目的地。主要功能如下：

1. 通信链路的建立与维护。
2. 系统管理信令的接收与发送。
3. 信令接收缓冲功能。
4. 业务应用注册处理。

单板系统管理模块通信设计点如下：

1. 单板系统管理模块启动时向系统管理服务器发送注册信息，建立于系统管理服务器之间的关系。
2. 单板系统管理模块定期向系统管理服务器发送心跳消息，根据系统管理服务器返回的响应消息，来确保与系统管理服务器之间的通讯畅通。
3. 单板系统管理模块在信令传递方面，采用单向滑窗机制，以保证信令传输的传输与可靠性。

## 数据模型

### 数据模型

本章建立的数据模型，是基于‘系统管理平台全面干涉应用数据’这一假设，包括数据解析和拼装两个过程。

‘全面应用数据干涉’的原因如下：因为系统管理模块自身也需要一些配置数据，需要解析和拼装。如果能设计一个通用的数据模型，能表达应用业务的需求，采用通用的数据模型来来表示相关业务数据，系统管理平台可以以通用的机制进行处理，而不需要区分数据关注的对象。

#### 表格化数据

二维表格化数据是关系型数据库的重要约定，某种意义上讲，任何数据都能用二维表格数据来表达。

数据模型表格化的另外一个重要目的是为了操作的模型化，对于表格，一般只有四类操作：插入、修改、删除、查询。这不仅仅是配置管理，还包括状态管理、故障事件管理以及应用控制功能都可以用表格化数据来表示。

按照这种模型，信令上就非常简单，总体上就只有四个业务操作信令，这对于信令的设计与平台的稳定都是非常利好的事情。最难模型化的业务是纷繁复杂的‘应用控制功能’，而这一块早在SNMP信令中得以体现和解决，SNMP定义的MIB数据规范中，使用‘control’类型的字段表示‘应用控制’的各类动作，其它字段表示该动作所需参数。通过这种方式也就把‘应用控制功能’映射到表格的‘Update’操作上了。

##### 表格化.简单数据

实际应用中存在简单的非表格配置数据，如像INI文件配置一样：

[ShellClassInfo]

IconIndex=0

IconFile=C:\Program Files\Thunder Network\Thunder\Program\downloadfolder.ico

InfoTip=迅雷下载目录

这是‘迅雷下载’的配置文件，它的数据很简单，并不是二维的数据。对于这类数据也可以二维表格化，只需要约束表格ShellClassInfo的记录数有且只有一条即可，即单实例表格。

##### 表格化.树状数据

关系型数据库在表格化树状数据时，采用主从关系模型来表达【主键与外键】。当然关系型数据库的关系不只是主从关系。

树状数据用主从关系展开后，对于从表(或细表)还可以有更强的能力支持(如从表上实现多索引)，但是失去了主从操作的直观性。

**本文档所提供的设计是基于树状关系来设计的，且没有完全展开为纯关系型模型。原因如下：**

1. **传统网管标准，包括SNMP/TR069/LDAP所支持的数据模型也只是树状模型(其中SNMP是全展开，TR069和LDAP都直接用树表达)。**
2. **作为管理型数据确实存在树状特征，而纯关系型在表达树状模型上，不够直观。**

**即便有更强烈的关系性需求，也可通过上层应用自行关联。**

#### XML模型抉择

在数据通信和数据的文件存储方面，将全面采用XML格式。然而XML从诞生到现在，已非常庞大，也有很多分支，我们该如何选择XML数据模型呢。

首先我们从基本的XML语义要素上分析，如下：

1. 头部分：
   1. 非实际数据内容。
   2. 主要描述一些版本，规范、语言等信息。
2. 节点
   1. 是实际数据内容。
   2. 节点(可称之为对象)可包含属性、元数据、子节点三个部分，如：

<person id="person\_id\_01">

<stuff\_id>1</stuff\_id>

<address size="32">shenzhen</address>

<age>25</age>

<gender>0</gender>

<url>www.google.com.hk</url>

</person>

其中id和size都是属性

stuff\_id、address、age、gender、url是节点person的子节点。

1、shenzhen、25、0、[www.google.com.hk](http://www.google.com.hk)是相应节点的元数据。

在描述数据时，可以选择属性或元数据，但没有强制的约定到底用哪个。一般说来，属性用于表达简单的、可预见的数据，元数据用以表达复杂的或不可预见的数据。

标准化的结构化数据描述有XPoiner、XPATH、XSD、XML-DB等。这里大部分都基于元数据的描述方法（因为他们企图描述通用的数据模型）。

XSD比较易用，可让应用层面关注较小，也正是因为如此，有时反而不好用，例如凡是结构化数据的翻译工作，都需要自动转化为一个函数实现，那么当我们需要新定义一张应用的配置表格时，是需要新作代码的，这对平台来讲，新业务的开发是很不利的。所以本文档不选择XSD作为描述规范。

另外XSD也基于XPATH/XPointer进行数据访问，XPATH/XPointer的访问规约是通过下标进行。如，“成人[1]. 孩子[3]”，不能基于条件查询，定位工作由应用自行计算好，这是有问题的。

在以往的实践中发现过较为严重的问题，就是当两个客户端同时访问一张表格时，如果其中一个客户端发起了对表格记录的删除，会导致另外一个客户端的访问错乱，对表格加锁对系统带来的设计难度是很大的，现实运行中也是容易出问题的地方。

XML-DB没有一个标准的语法规范，也没有一个标准的独立实现，都是和自己的数据库系统绑定的，不同数据库厂商略有不同，大体上都是利用元数据部分传递SQL语句的核心部分，利用属性提供XPATH信息。

**我们的数据模型其实并不需要很复杂，本文档的设计中仅仅利用了节点和属性两个元素就能满足了，也鉴于上述分析，我们可以自行设计XML的数据模型。**

##### 表格模型描述

本章节所描述的表结构信息，只是一个模型，并不存在真实的XML样式。

本章所描述的表格模型，企图能涵盖：配置数据、故障事件数据。

##### TableInfo

表格的最顶端定义，由XML节点标记TableInfo括起来，即：

<TableInfo >

<!-- Field Info -->

<!-- Index Info -->

<!-- Detail Table Info -->

</TableInfo>

TableInfo节点的属性定义：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Name | string | M | 描述表名，需要符合文件名规范，可不分大小写（对应的文件名全小写）。 |
| Single | bool | O | (1)允许的取值范围为：true, false，缺省为false  (2)用以表示该表格是单实例表还是多实例表。 |
| MaxInstance | uint32 | O | 当多实例表时，用以指定最大记录数。缺省不做最大记录数限制。多实例表最大记录数为0时，表示不存储 |
| OverFlowPolicy | enum | O | 当多实例表有最大记录数限制时，该表数据满额后的Insert策略定义，现预定义如下：  0=失败；  1=删除最先插入的记录  2=删除根据时间字段显示的最老记录 |
| TimeField | string | O | 当OverFlowPolicy为2时，指定的时间字段名。 |
| Dynamic | enum | O | 取值范围：fixed, static, dynamic, 仅仅限制客户端是否允许动态增删记录，缺省为dynamic。   1. 固定表格(fixed)：这类表格需要必须要有出厂数据，在现场是不允许进行增删记录。 2. 静态表格(static)：在相关应用或服务模块运行期间，不允许增删。 3. 动态表格(dynamic)：在相关应用或服务模块运行期间，允许增删。 |
| DbFlag | bool | O | 写数据库标志，true, false，缺省为false |
| SDN | int | M | 同步范围定义，取值-1，0，1，缺省为0  -1表示该表不参与同步，  0表示在同一管理范围内同步  1表示跨管理范围同步 |

1. 细表是主表记录的属性扩展，多实例表必须有索引，如果应用确实没有可索引的字段，也需要添加辅助的索引字段。以此支持树状的数据表达。
2. 数据库表格不能是细表，也不支持它的细表。数据库中的表格采用单独的SQL脚本创建。
3. SDN，请参考数据同步一章节的设计。

##### FieldInfo

<TableInfo> … </TableInfo >内可包含多个字段描述，每个字段由<FieldInfo/>表示。

FieldInfo节点的属性定义为：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Name | string | M | 描述字段名，可不分大小写。 |
| Type | enum | M | string, 普通文本  name，定义一种标识符规范的文本数据  filename，定义一种满足文件名规范的文本数据  int8,int16, int32, int64,uint8,uint16,uint32,uint64，整型  bool, 取值范围为'1'、'0'、'true'、'false'、'yes'、'no'  enum，整型的枚举类型  datetime UTC时间格式的字符串，**如果第一阶段不好实现，可不考虑该类型**。  base64, base64编码串  binary, 二进制数据串 |
| case | bool | O | (1)仅针对文本型数据有效（string & name）,缺省为true  (2)True:大小写敏感；false：大小写不敏感。 |
| Range | string | O | (1)缺省不做范围限制。  (2)由{}括起来的多个范围组合，范围之间用逗号分隔  (3)范围可由冒号分隔，如:  (a)a:b，表示最小为a，最大为b  (b)a:，表示最小为a，无最大限制  (c) :b，表示最大为b，无最小限制  (d) c，表示为单值c，即c:c的简写，enum的枚举项只能是单值  (4)string, base64,binary的范围表示长度范围  (5)整型的范围表示数值的取值范围。  (6)datetime表示时间范围 |
| NotNull | bool | O | 1. 表示该字段是否允许为空。缺省为false； |
| Default | string | O | 1. 指定缺省值时，一定不允许为空(忽略NotNull)。 2. 当插入新纪录没有指定该字段值时，需要用默认值。 |
| Dynamic | enum | O | 取值范围：read-only, static, dynamic，仅仅限制客户端是否允许动态修改。缺省为dynamic   1. 只读字段(read-only)，表示该字段是只读字段，不可以修改。 2. 静态字段(static)，在相关应用或服务模块运行期间，不允许修改。 3. 动态字段(dynamic)，表示该字段的修改可实时生效。 |
| Storage | enum | O | 取值范围为：persistent, temporary，缺省为persistent   1. 持久性数据(persistent)，需要存储于文件或数据库中，系统重启能恢复的数据。 2. 临时性数据(temporary)，仅存在于内存中，系统重启后就丢失了，可重新获取，往往状态性数据就是这样的。 |
| Location | enum | O | 取值范围：system、application，缺省为system   1. 系统侧数据(system)，表示数据来源为系统管理平台。 2. 应用侧(application)，表示数据来源为应用侧。 |
| Primary | bool | O | 1. 用以指示主键字段，一张表可以含由多个主键字段构成唯一主键索引。 2. 主键字段是只读的，并不能为空。对于主键的修改由客户端发起先删后插的流程完成。 3. 缺省为false |
| Sync | bool | O | 1. 指示该字段变更，是否需要同步到对端。 2. 缺省为真，表示需要同步。 |

1. 对于实际字段的取值范围，可能还会有更进一步的语义要求或上下文要求，如：
   1. 满足某种格式的字符串，甚至这种格式可定义，像正则表达式一样。
   2. 根据其他数据(可能来自其它表格)的组合、或部分、或变换等。

对这类要求，约定由客户端实现，在系统管理平台不做约束。

1. Sync，请参考数据同步一章节的设计。
2. 关系型数据库系统，仅允许定义一个主键字段，而网管系统是可以定义多个主键字段的。

#### 系统目录的规划

系统管理平台在运行过程中，共有以下几类文件需要保存：

1. 配置数据文件
2. 软件文件
3. 运行时产生的各种临时文件

为了避免冲突，各种文件最放在不同的目录下，以防造成相互的影响。

本文档对文件保存路径做以下约束：

1. 文件保存目录设置如下

—AppHome

| |—SSN+MOI

| | |—APP

| | |—-CFG

| | |—-TMP

| |—……

—SmpHome

| |—SMS

| | |—APP

| | |—CFG

| | |—TMP

| |—BSM

| | |—APP

| | |—CFG

| | |—TMP

1. AppHome\：存放业务应用软件相关文件的根目录。
2. AppHome\SSN+MOI\：业务软件相关文件保存路径，业务软件由SSN+MOI确定。
3. AppHome\SSN+MOI\APP\：用于存放业务软件本身的相关文件。
4. AppHome\SSN+MOI\CFG\：用于存放业务软件相关配置文件。
5. AppHome\SSN+MOI\TMP\：用于存放业务软件产生的临时文件或运行时文件。
6. SmpHome\：存放系统管理平台相关文件的根目录。
7. SmpHome\SMS\：存放系统管理服务器相关文件。
8. SmpHome\SMS\APP\：用于存放系统管理服务器软件自身的相关文件。
9. SmpHome\SMS\CFG\：用于存放系统管理服务器使用并保存的相关配置文件。
10. SmpHome\SMS\TMP\：用于存放系统管理服务器产生的临时文件或运行时文件。
11. SmpHome\BSM\：存放单板系统管理模块的相关文件。
12. SmpHome\BSM\APP\：用于存放单板系统管理模块软件自身的相关文件。
13. SmpHome\BSM\CFG\：用于存放单板系统管理模块需要读取的相关配置文件。
14. SmpHome\BSM\TMP\：用于存放单板系统管理模块产生临时文件或运行时文件。

#### 统一编码

##### 管理范围编码

使用MDN(Manage Domain Number)管理范围编号为系统管理服务器进行逻辑编号。约定0xFF为无效值。

一个系统管理范围允许部署多个系统管理服务器，但是都拥有相同的系统管理编号，即从逻辑上只有一个系统管理服务器。

##### 单板系统编码

使用MUN(Manage Uni-board Number) 单板系统编号为单板系统管理模块进行逻辑编号。一个系统管理范围可管理多个单板系统管理模块。约定0xFF为无效值。

单板系统管理模块自身也是一个应用，也需要按照应用系统编号法则进行编号。

##### 应用系统编码

使用SSN(Sub System Number) 子系统编号为某类应用进行逻辑编号。约定0xFF为无效值。SSN是一种系统资源，需要集中统一管理，每开发一类新的应用，都需要分配一个SSN。

##### 对象实例编码

使用MOI(Managed Object Identity)管理对象标识为某个具体应用的逻辑编号。约定0xFF为无效值。MOI是针对于每一个应用软件实例分配的编号。在一个子系统类型下，MOI必须是唯一的，不允许重复。

在实际的网络部署中，我们可以用SSN+MOI来表示一个实际部署的管理对象，并用MDN+MUN来表示管理对象的位置。

#### XML-SQL

本节定义基于XML的结构化查询语言，需要说明的是，国际标准中虽然有XML-SQL提法，但没有统一的规范，不同数据系统厂商，提供的实现都略有不同。

这里根据系统管理自身的特征及要求，定义自己的规范，下表预定义几个概念：

|  |  |
| --- | --- |
| 概念 | 说明 |
| 语句 | 定义了Insert、Update、Delete、Select、Result等5种语句 |
| 子语句 | 5种语句可以嵌套自身的语句称为子语句，这是为了表达树状的数据模型。 |
| 子句 | 定义了Set、Where、Record、Field四种子句。 |
| 叶子语句 | 不含子语句的语句，以及最底层的子语句称为叶子语句。 |
| 分支语句 | 把含有子语句的语句称为分支语句。 |
| 根语句 | 不含子语句的语句，以及最顶层的分支语句称为根语句。 |

##### Insert语句

<Insert>

<!--

Field子句，

如果该子句在Insert分支语句中，仅包含所有主键字段，用于定位细表。

如果该子句在Insert叶子语句中，可以包含各种字段，用于构建新记录。

-->

<!--

可选Insert子语句，如果操作的是细表则需要包含Insert子语句，结构上和Insert语句类似。

-->

</Insert>

<Insert>节点属性定义

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Table | string | M | 描述表名， |

样例：

<insert table=” person”>

<field name=” stuff” value=” 1”/>

<field name=” address” value=” shenzhen”/>

<field name=” age” value=” 25”/>

<field name=” gender” value=” 0”/>

<field name=” url” value=” www.google.com.hk”/>

</insert>

##### Update语句

<Update>

<! --

Set 子句，该语句仅存在于Update叶子语句，用以表示需要修改的字段。

-->

<!--

Where 子句，当该子句存在于Update叶子语句，则用以定位修改记录。

-->

<!--

Field子句，当该子句存在于Update分支语句中，仅包含所有主键字段，用于定位细表。

-->

<!--

可选Update子语句，如果操作的是细表则需要包含Update子语句，结构上和Update语句类似。

-->

</Update>

< Update >节点属性定义

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Table | string | M | 描述表名， |

样例：

<update table=” person”>

<set>

<field name=” address” value=” shanghai”/>

<field name=” age” value=” 30”/>

</set>

<where>

<field name=” stuff” value=” 1”/>

</where>

</update >

##### Delete语句

<Delete>

<!--

Where 子句，当该子句存在于Delete叶子语句，则用以定位修改记录。

-->

<!--

Field子句，当该子句存在于Delete分支语句中，仅包含所有主键字段，用于定位细表。

-->

<!--

可选**Delete**子语句，如果操作的是细表则需要包含**Delete**子语句，结构上和**Delete**语句类似。

-->

</Delete >

< Delete >节点属性定义

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Table | string | M | 描述表名， |

样例：

<delete table=” person”>

<where>

<field name=” stuff” value=”2”/>

</where>

</delete >

##### Select语句

<Select>

<!--

可选Where 子句，当该子句存在于**Select**叶子语句。用以定位查询记录。

如果没有该子句，表示查询所有记录。

-->

<!--

Field子句，当该子句存在于**Select**分支语句中，仅包含所有主键字段，用于定位细表。

-->

<!--

可选**Select**子语句，如果操作的是细表则需要包含**Select**子语句，结构上和**Select**语句类似。

-->

</Select>

< Select >节点属性定义

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Table | string | M | 描述表名， |
| Filter | string | C | 仅存在于Select叶子语句中。用于表示查询的字段:  (1)\*表示所有字段；（缺省值）  (2)多个字段用逗号分隔。 |

样例：

<select table=”person” filter=”suff,age, gender”>

<where>

<field name=” age”, value=”50” compare=”ge”>

</where>

</select

##### Result语句

该语句用于表示Insert、Update、Delete、Select语句执行的结果。

<Result>

<!—

可选Record 子句，用于表示Select返回的结果集

-->

</Result>

Result属性定位

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Code | uint32 | M | 0=操作成功；1=系统正忙；2=其它错误；3=认证失败；4=表不存在；5=定位失败；6=不允许操作；7=无效数据；8=插入冲突；9=容量满额。  Insert：0~9；Update：0~7；Delete：0~7；Select：0~7 |
| Count | uint32 | C | 可选属性，用于表示Select查询返回结果集的记录总数。  该语句嵌入的具体消息可分多次返回。每次返回可按单条或多条记录返回。 |

Select的Result样例

<result code=”0” count=”10”>

<record>

<field name=” stuff” value=” 100”/>

<field name=” age” value=” 50”/>

<field name=” gender” value=” 0”/>

</record>

</result>

<result code=”0” count=”10”>

<record>

<field name=” stuff” value=” 101”/>

<field name=” age” value=” 52”/>

<field name=” gender” value=”3”/>

</record>

</result>

##### 子句

###### <Set>…</Set>子句

<Set>

<!--

Field子句，用以表示需要修改的字段。

-->

</Set>

该语句用于Update语句中。

###### <Where>…</Where>子句

<Where >

<!--

Field子句，在Update&Delete语句中，仅包含主键字段值，以限制操作唯一性定位。

在Select语中可以携带任何字段，以支持多条记录的查询。

-->

</Where >

如果需要批量修改或删除数据，由客户端自行设计。

###### <Record>…</Record >子句

<Record>

<!—

Field子句，这里不包含主表的主键字段

-->

</Record >

###### <Field/>子句

该子句节点没有子节点，其属性定义为：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Name | string | M | 描述字段名。 |
| Value | string | M | 描述字段值。 |
| Compare | string | O | 1. 仅用于Select的Where子句 2. lt，le，eq，ge，gt 3. 缺省为eq |

如果用在Result语句中，不包含主表的主键字段。

#### 数据的文件存储

##### 数据存储总体方案

###### 公共配置数据的存储

一个管理范围内需要能管理自治，那么跨管理范围的公共配置理应在不同管理范围内独立存储。在同一管理范围内的多个服务器上，也要分别存储公共配置数据。

###### 配置数据按照配置表进行独立存储

不同的配置表数据存储于不同文件，这样某个表的变更，只会影响单个文件的变更，可降低对文件IO的影响。

(1)我们可以按照SSN+MOI的方式组建不同的目录，将不同应用软件实例的配置文件放到相应的目录。这样可以做到应用软件之间的配置数据不会相互影响。相同类型的应用软件的不同实例同样也可以区分开。

(2)在每一个应用软件实例的配置文件目录下，我们创建不同的文件，分别存放不同配置表的配置数据，并按照表名或者表ID的方式命名对文件进行命名。

##### XML文件格式

###### <Table> … </Table>

特定表格的数据用<Table> … </Table>括起来，即：

<Table>

<!-- Record -->

</Table>

<Table>节点的属性定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Name | string | M | 描述表名，需要符合文件名规范，可不分大小写（对应的文件名全小写）。 |

1. 只要某张表格的数据具有persistent类型的字段，该表格就需要文件存储。
2. 如果主表不满足文件存储要求，细表也不会被存储。
3. 文件名为” $AppHome\$SSN+MOI\CFG\TableId.xml”

###### <Record> … </Record >

表格中的记录由<Record> … </Record >括起来，一张表格可以包含多个<Record>子节点，即：

<Record>

<!—Field Data -->

<!—Detail Table Data -->

</Record >

因为按表分别存储，为了正确加载数据，从表中也要记录主表的主键字段，也从根往叶子方向嵌套存储。

<Record>节点的属性定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Time | datetime | M | 最后一次修改时间 |

该属性为后续的一致性检查做铺垫，可考虑在后续实现。

###### <Field/>

< Field>节点的属性定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 选项 | 属性说明 |
| Name | string | M | 描述字段名。 |
| Value | string | M | 描述字段值。 |

1. 只有persistent类型的字段，才会被存储。
2. 字段为空时，不存储。

##### 内建数据字典

$AppHome\$SSN+MOI\CFG\TableInfo.xml

$AppHome \$SSN+MOI\CFG\FieldInfo.xml

这两个文件放在AppHome目录下的原因是这两个文件，是运行期不变的，除非版本升级。这两张表归系统管理服务器和单板系统管理模块使用。

其中SSN+MOI用于标识唯一业务应用实例，数据字典的修改，可随应用发布，系统管理平台可以不做任何的发布工作。

TableInfo&FieldInfo定义该应用的所有表的数据字典，而TableInfo&FieldInfo本身也是表格，但TableInfo&FieldInfo自身的数据字典采用内建的方式。

数据字典系统表TableInfo的内建定义约定如下：

<TableInfo Name=” TableInfo” >

<FieldInfo Name=”TableId” type=”int” Index =”true”/>

<FieldInfo Name=”TableName” type=”name” case=”false”/>

<FieldInfo Name=”Single” type=”bool”/>

<FieldInfo Name=”MaxInstance” type=”uint” />

<FieldInfo Name=”Dynamic” type=”enum” enum=”{fixed=0, static=1, dynamic=2}”/>

<FieldInfo Name=”DbName” type=”string” />

<FieldInfo Name=” BoardStroage” type=”bool” />

<FieldInfo Name=”SDN” type=”int” range=”{-1:65535}” default=”0”/>

</TableInfo>

数据字典系统表FieldInfo的内建定义约定如下：

<TableInfo Name=”FieldInfo” >

<FieldInfo Name=”TableId” type=”int” Index =”true”/>

<FieldInfo Name=”FieldId” type=”int” Index =”true”/>

<FieldInfo Name=”FieldName” type=”name” case=”false” NotNull=”true” />

<FieldInfo Name=”Type” type=”string” NotNull=”true”

enum=”{string,name,int,uint,long,ulong, bool, datetime,base64,binary}”/>

<FieldInfo Name=”case” type=”bool”/>

<FieldInfo Name=”range” type=”string”/>

<FieldInfo Name=”enum” type=”string”/>

<FieldInfo Name=”notnull” type=”bool”/>

<FieldInfo Name=”Default” type=”string”/>

<FieldInfo Name=”Dynamic” type=”enum” enum=”{read-only=0, static=1,dynamic=2 }” />

<FieldInfo Name=”location” type=”enum” enum=”{system=0, application=1}” />

<FieldInfo Name=”storage” type=”enum” enum=”{persistent=0, temporary=1, control=2}” />

<FieldInfo Name=”Primary” type=”bool”/>

<FieldInfo Name=”Sync” type=”bool” default=”true”/>

</TableInfo>

系统启动时可根据TableInfo&FieldInfo，创建应用表格。

该数据字典的维护可通过特定的客户端软件实现，特定客户端发起通用的表格操作来进行，系统管理平台对数据字典的维护支持应处于特定情况下，例如版本发布、版本升级等。在平时不允许对数据字典进行维护。

#### 数据的数据库存储

详细设计文档，做详细设计说明，这里只做要求，提供一个通用的API接口，用于数据库操作，包括四个数据操作（Insert, Delete, Update, Select【需要支持多批返回】）和两个字典维护操作Create， Drop等。Create用于创建数据库、表格、索引、视图等，需要考虑对象已存在，并自动进行修改的能力支持。

下层对接的数据库，可先以SQLITE为例。

### 通信模型

#### 组网模型

##### 子网划分



图4‑4系统管理平台通讯模型子网划分

以系统管理服务器为中心，把涉及到与之通信的网络实体的集合定义为子网。

根据通讯对象与类型的不同，我们将通讯模型划分三个子网，客户端接入并与系统管理服务器通讯的子网称为客户端访问通信子网；同一管理范围内的子网称为管理范围内通信子网；管理范围之间相互的通信的子网称为管理范围间通信子子网。

单板系统(Uni-System)和客户端仅处于一个子网中，系统管理服务器分别处于三个子网中。

子网划分的目的：

(A).客户端的安全访问和正确寻址。

(B).减少数据同步量。

(C).访问的简单性。

##### IP划分

###### 客户端的IP

即客户端所在主机IP地址，即SC-C-IP，其中SC表示为客户端访问通讯子网。

###### 单板系统的管理IP

单板设备上分配的管理IP地址，即SB-B-IP，其中SB表示为管理范围内通讯子网。

###### 系统管理服务器的管理IP

由于系统管理服务器跨三个子网，因此从通讯模型上来说，需要为系统管理服务器分配三个管理IP地址，分别为SC-S-IP、SB-S-IP、SS-S-IP。，其中SS表示为管理范围间通讯子网

###### IP子网关系



图4‑5通讯子网IP分配关系

如图所示：

1. 客户端IP地址为SC-C-IP，客户端通过该IP地址与系统管理服务器1进行通讯，SC-C-IP属于客户端访问通讯子网的内部IP地址。
2. 单板系统管理模块1、2的IP地址分别为SB-B-IP1和SB-B-IP2，单板系统管理模块通过该地址与系统管理服务器1进行通讯，SB-B-IP1和SB-B-IP2属于管理范围内通讯子网1的内部IP地址。
3. 系统管理服务器1拥有3个IP地址，分别是SC-S-IP、SS-S-IP1和SB-S-IP1。
4. SC-S-IP为客户端访问通讯子网IP地址，系统管理服务服务器通过该地址与客户端在客户端访问通讯子网中进行通讯；
5. SS-S-IP1为管理范围间通讯子网的IP地址，系统管理服务器1通过该地址与其他系统管理服务器在管理范围间通讯子网中进行通讯；
6. SB-S-IP1地址管理范围内通讯子网1的IP地址，系统管理服务器1通过该地址与管理范围内的单板系统管理模块在管理范围内通讯子网1中进行通讯。
7. 在逻辑上，每个管理范围都有其相应的管理范围内通讯子网，如上图的管理范围内通讯子网1、2。管理范围内通讯子网间并不重叠。

###### IP分配方案

（1）对于客户端访问通讯子网中的IP地址，客户端侧可以直接通过用户手动配置所在操作系统的IP地址。系统管理服务器侧可以采用静态配置的方式，由系统管理服务器读取配置信息，获取IP地址信息。

（2）对于管理范围间通讯子网中的IP地址，系统管理服务器可以通过读取本地硬件信息（例如机框号），按照一定生成规则，生成响应的IP地址。

（3）对于管理范围内通讯子网中的IP地址，可通过DHCP协议进行动态分配。

#### 可靠通信

##### 网络承载协议选择

在网络承载协议方面，有很多协议可供选择，如TCP、UDP、SCTP、IPX、SPX等网络协议。为了降低系统要求，建议在最普通的TCP和UDP之间选择。

选择TCP的好处在于自动实现传输的可靠性，不好的地方在于需要管理连接，针对每个连接需要独立的接收线程，比较浪费资源，当然也可以采用异步机制，但处理过程比较复杂。

UDP协议的优点是网络模型简单，实现容易，不需要建立和管理连接，资源消耗少，缺点是无法保证传输的可靠性。但是可以通过增加额外的保障手段，来提高传输的可靠性。

基于之前的设计决策，系统管理平台决定采用UDP协议进行通讯。

##### 消息序列化设计

###### 序列化模型



图4‑6消息序列号模型

###### 序列化队列

对网络上接收到的消息按照接收的时序进行排队，是一个循环队列。上图其中0表示该序号对应的消息丢失，1表示该序号对应的消息未丢失，2表示再也取不回来消息了(将忽略处理)。

消息入队列机制，如果是新消息，放置队列尾端t，t向右移动。如果是旧消息，则填入相应位置（由滑窗控制识别消息的新旧）。

消息出队列机制，用上图说明，当头端h空消息处填充好后，立即启动消息出队列流程，弹出一个处理一个，h向右移动，知道h到达t或空消息位置。

消息的入队列和出队列由同一线程完成。

序列化消息队列节点记录消息的内容和发端节点编号。

###### 接收滑窗

对某个节点的接收消息进行按消息序号排队，滑窗节点项记录消息属性：(1)是否为空，(2)对应在序列化队列的位置。

接收端滑窗记录头/尾端的消息序号。接收到某条消息，需要判断该消息是否过期，还需要判断某条消息是否为新消息，若是新消息，与尾端消息相差多少个丢包消息。

当序列化队列头端右移时，根据该消息发端节点编号，对相应接收端滑窗进行头端右移。

滑窗重要参数

nSize，表示滑窗大小。

nHead，表示滑窗的头部消息序号。取值范围：[ 0，nSize-1 ]。

nTail，表示滑窗的尾部消息序号(下一个应收的消息)。取值范围：[ 0，nSize-1 ]。

bFirst，表示系统没有放入过消息。

滑窗存储的消息范围为：[ nHead，nTail )。

初始：nHead = 0；nTail = 0；bFirst=true；

收到第一个序列化消息时，会执行下述逻辑。

if(bFirst)

{

nHead = nSeq;

nTail = nSeq;

bFirst = false;

}

消息存放位置算法

nDistance = nSeq % nSize

其中nSeq表示当前消息的序号。

nDistance表示应该存储在滑窗中的位置，如果用数组实现滑窗oWindow，那么oWindow [nDistance]就是代表消息存放所在的内存空间。

消息过期算法

nDistance1 = nSeq - nHead;

if(nDistance1 >= nSize)

{

if(nDistance1 <=( 0xFFFFFFFF - nSize + 1))

{

//可能是系统重启或长时间断链导致。

//滑窗异常处理，继续后续流程。

}

else

{

//消息过期，可丢弃当前消息，中断当前流程。

}

}

新消息与旧消息的判断

nDistance2 = nTail - nHead;

if(nDistance1 >= nDistance2)

{

//表示nSeq对应的消息是新消息

//nSeq - nHead表示应该插入的空消息数。

//如果空消息插入序列化队列失败(满了)，需要启动滑窗异常处理。

}

else

{

//表示nSeq对应的消息是旧消息

//nSeq% nSize表示消息应该存于何处。

}

滑窗异常处理

while(nHead!=nTail)

{

nDistance = nHead % nSize

if(oWindow[nDistance]是空消息)

{

//那么在序列化队列的相应位置也应该是空消息

//应该认为该空消息再也取不回来了，设置为2，让系统继续能处理。

}

++ nHead;

}

bFirst = true;//很重要，表示下次收到消息，将重新开始。

除了消息序号在可承受范围内过期需要进行滑窗异常处理外，收到注销消息，也需要进行滑窗异常处理。

###### 保持队列

当开启序列化服务后，允许接收序列化消息，但是由于某种原因，致使上层应用模块还不能真正成为服务状态时，序列化消息出队列后，需要放入保持队列中，当上层应用模块真正成为服务状态时，才开始序列化消息的处理。

直到保持队列为空时，序列化消息即转入即时处理。当然该队列是可选的。

###### 发送队列

发送队列中的消息分两类：群发消息、非群发消息。使用目标节点号来分别，-1表示群发消息，否则为具体目标的非群发消息。

由于有非群发消息，所以为每个目标节点单独维护消息序号，用发送序号队列表示（类似于接收窗口）。

实际设计时，需要支持两种模式，一种仅支持群发的模式，一种支持非群发模式。

###### 消息接收流程

收到的消息，从消息中提取发送方身份NodeId，并将序号信息放入相应的接收滑窗。

(A)过期包(序号超出队列范畴)，则丢弃。

(B)重复包(序号已在队列中，且有非空标记)，则丢弃。

(C)丢包(序号已在队列中，且有空标记)，则根据位置信息，直接把消息数据存入序列化队列。

(D)正常包(序号不在队列中，且未超出队列范畴)：

(1)该序号与节点队列的最后一个消息的序号有空隙，表示有丢包，则插入空消息到序列化队列(返回位置信息)，并将位置信息存入节点队列，做空标记。

(2)若无丢包，将消息存入序列化队列(返回位置信息)，并将位置信息存入节点队列，做非空标记。

###### 消息重传流程

定时扫描接收端滑窗，提取空消息标记，用单播模式向该节点发送端发送消息重传请求。发送端收到消息重传请求时，从发送队列提取相应消息，并用单播模式重发该消息。

###### 异常流程

异常发生时，序列化队列将从头到尾的处理消息【忽略空消息】。有以下几种异常情况：

(A)接收端滑窗或序列化队列满。

(B)发现某个节点重新注册，而该节点的接收滑窗还有消息。

(C)某节点长时间断连。

再好的序列化方案都只能保证一定时间网络断链情况，网络恢复正常后，系统的数据能一致起来。

长时间断链情况，可作为特殊情况对待，演进版本可考虑一致性检查和恢复机制。为预留这个机制，我们将在数据模型的设计上增加时间信息，一致性恢复时将以最新时间为依据。一致性检查出差异报告，可支持自动恢复和人工恢复两种模式，自动恢复的原则是：即便数据错误，也要让整个系统的数据错成一样的。

###### 定时流程

有三个定时流程。

(A)定时注册，建议1秒钟一次注册。

(B)定时检查丢包情况，发起消息重传流程。

(C)定时断线检测，发起异常处理流程。

##### 令牌模型设计

通信模块子网中有时为了某种需要，需要独占时间，进行某种事务的处理。本令牌设计模型实现1+m的能力，为了支持n+m的能力，我们设计多令牌支持方案，即n个令牌就实现了n+m的能力，这里n+m总数就是子网节点数。

这里描述令牌模型，主要是为了实现在同一管理范围内部署的多个系统管理服务器中，竞选出一个服务器对外进行服务，从而展现出一个逻辑服务器。

本节所描述的设计方案依据RFC5059标准。

###### 令牌结构设计

|  |  |  |
| --- | --- | --- |
| 参数名 | 数据类型 | 参数说明 |
| TokenId | Uint32 | 令牌标识。 |
| DefaultNode | Uint32 | 缺省由那个节点占用。 |
| Nodes | strings | 用逗号分隔的参与竞选的节点列表。 |
| SDN | uint32 | 令牌竞选的同步范围。 |

令牌是一种资源，就像网络端口号一样，其编号必须统一定义和管理，以防不同模块之间的冲突。

###### 令牌对应用的接口。

令牌是一个自动化服务，根据配置自动启动。采用回调接口告知应用某个令牌被抢占或放弃。

###### 令牌竞选流程



图4‑7令牌竞选流程

其中四个节点CPEA表示四个竞选状态（P为初始状态），有向连线表示状态迁移的驱动消息。

###### 竞选状态

|  |  |
| --- | --- |
| 状态 | 状态描述 |
| C | Candidate 候选状态。本节点是候选者。此时另外一个节点是占有令牌。 |
| P | Pending 等待状态。等待选举结果，此时本节点认为无其他节点占有令牌，但本机还未当选。 这个临时状态是为了防止竞选时出现振荡。 |
| E | Elected挂职状态。节点已当选，但未正式就职的准备状态，这个状态是为了原令牌节点做清理。 |
| A | Acknowledged就职状态。表示本节点正式占有令牌。 |

###### 驱动消息

p=Preferred BSM; 更优BSM消息

n = Non-Preferred BSM非最优BSM消息

n\*= Non-preferred BSM from Elected Owner当选Owner的非最优BSM消息

T= Bootstrap Timer timeout超时消息，不同状态下timer值可不同。

BSM(Bootstrap Message)消息定义：其实只有1条消息，但是根据消息发送者的weight和接收者的状态、接收者自己的weight、以及当前Owner的weight的对比关系，可以分为（每个节点都要记录当前Owner的地址信息和weight）：

P状态： 若‘发送者权重’> ‘自身权重’，则‘P消息’，否则‘N消息’

A、E状态：若‘发送者权重’> =‘Owner权重’，则‘P消息’，否则‘N消息’

C状态：

(1)若‘发送者权重’> =‘Owner权重’，则‘P消息’，否则：

(2)若‘发送者权重’> =‘自身权重’，则‘P消息’，否则：

(3)若‘发送者 == Owner’，则‘N\*消息’，否则‘N消息’

###### 状态签转

C候选状态 (Candidate)

1. 收到Preferred BSM 消息：

* 状态不变。
* 存储Owner信息；
* 设置BST定时器为 BS\_timeout

1. 收到BST超时消息：

* 转P状态，老Owner没动静了，我要参选！
* 设置BST定时器值为 BS\_rand\_override。

1. 收到n\*消息

* 转P状态 ，老Owner体力不支了(weight变小了，这种情况下其他成员也都知道了)，或者本机自身的weight提高了(这种情况只有本机自己知道)，我要参选！
* 设置BST定时器值为(BS\_timeout + BS\_rand\_override)。

1. 收到n：（RFC中没有定义处理行为）

* 状态不变

P等选状态（Pending）

1. 收到Preferred BSM 消息：

* 转C状态
* 存储Owner信息；
* 设置BST定时器为 BS\_timeout

1. 收到BST超时消息（BS\_timeout）：

* 转E状态，当选，但不正式‘就职’
* 发送 BSM消息。
* 设置BST定时器值为 BS\_prepare。

1. 收到Non-preferred BSM：（RFC中没有定义处理行为）

* 状态不变。

E挂职状态 (Elected)

1. 收到Preferred BSM 消息：

* 转C状态 – 比自己更优的组长来了，立刻让位…
* 存储Owner信息；
* 设置BST定时器为 BS\_timeout

1. 收到BST超时消息：

* 转 A状态，Acknowledged，正式就职
* 发送 BSM消息
* 设置BST定时器值为 BS\_period。
* 将自身信息保存为Owner信息，并对应用产生竞选成功的回调。

1. 收到Non-preferred BSM：

* 状态不变
* 发送 BSM消息，有挑战者立刻镇压
* 原BST定时器不重置，继续计时

A就职状态 (Acknowledged)

1. 收到Preferred BSM 消息：

* 转C状态 – 比自己更优的组长来了，立刻让位…
* 存储Owner信息；
* 设置BST定时器为 BS\_timeout
* 对应用产生放弃令牌的回调。

1. 收到BST超时消息：

* 状态不变
* 发送BSM消息--该提醒一下各位了，我是组长
* 设置BST定时器值为 BS\_period

1. 收到Non-preferred BSM：

* 状态不变
* 发送BSM消息--不知轻重的小卒来挑战，立刻镇压

###### 竞选权重

每个节点对某个令牌设计的权重计算如下：

1、基础权重b-weight：当前节点是该令牌的缺省节点，那么b-weight=192，否则b-weight=64。

2、参考权重r-weight：用当前节点的节点号作为参考权重。

3、节点权重n-weight(uint64) =( ((uint64)b-weight)<<32) | r-weight。

###### 竞选定时

|  |  |  |
| --- | --- | --- |
| 编号 | 名称 | 说明 |
| T0 | bs\_refer | 参考时长（以秒为单位） |
| T1 | bs\_period | 当选的Owner周期性宣告。1.5\*T0 >= T1 >= T0 |
| T2 | bs\_timeout | 成员探测Owner不可达时长。建议T2=2\*T1+ T0 = 4\*T0 |
| T3 | bs\_prepare | 挂职等待时长，该时间用于原Owner做清理工作。建议为1.5\*T0 |
| T4 | bs\_rand\_override | 从C到P后设置的定时时长。weightDelay+ addrDelay+ T0。  weightDelay = (MaxWeight – b\_weight) 0.5\*T0/10  addrDelay = myIPAddress的第三字节 / 128\*T0/10 |

换句话说竞选定时仅需要一个参考时长的配置，该时长在测试阶段可作为临时配置，系统稳定后，根据测试情况固定一个值即可。

###### 默认调整

当网管发起命令进行默认抢占令牌的节点标识变化时：

1. 当自己之前是默认的，本调整为非默认的，需要更新。
2. 当自己之前不是默认的，调整为默认的，也需要更新。
3. 否则不做任何更新。

当自己的默认角色发生改变时，需要重新计算权重，和对该令牌的T4定时时长。

###### 异常处理

当网络出现异常时，可能会引发多个子网的出现，从而导致一个令牌同时被多个节点占用。这个问题并不会引起太大的影响，只要当网络恢复正常时，会有一个收敛时间（>=T2），该收敛时间一过，是可以重新竞选出新占用者的。

##### 虚拟IP设计

这里的虚拟IP，实际上也是真实的IP地址。基于令牌技术，获得令牌的节点可以绑定相应IP地址，失去令牌的节点需要解除相应IP地址的绑定。

## 执行方案

### 数据配置

#### 配置数据初始化

##### 业务设计

业务设计如下：

1. 系统管理服务器配置模块启动，读取基本配置数据文件，获取系统信息，了解系统构成。（主要包括系统公共配置信息，系统组成信息，了解系统构成，划分为那些管理范围，每个管理范围包含哪些单板，每个单板上有哪些业务应用）
2. 系统管理服务器根据系统配置信息，依次读取各个业务应用软件的配置数据文件，如果文件不存在，则报错。
3. 系统管理服务器将读取的各种配置数据保存在内存中，供配置更新使用。
4. 单板系统管理模块启动后，根据软件管理配置信息，依次读取本板上各个业务应用软件的配置文件。
5. 单板系统管理模块计算每个业务应用软件配置文件的MD5校验码，组织并发送校验信息到系统管理服务器。
6. 系统管理服务器收到校验信息，并与本地文件进行校验，如果校验通过，则发送校验结果。否则，将本地相应的配置文件更新下去。
7. 单板系统管理模块收到校验结果进行判断，如果校验通过，则读取并解析本地的配置数据文件内容，并将其保存在内存中。
8. 如果校验结果不一致，则单板系统管理模块用系统管理服务器更新下来的配置文件替换本地配置文件，并重复步骤7的相关操作。

##### 业务流程

图4‑8配置数据初始化流程图

业务流程描述：

系统管理服务器侧

1. 系统管理服务器配置模块启动。
2. 系统管理服务器读取基本配置数据文件，获取系统信息，了解系统构成。
3. 系统管理服务器依次读取各个业务应用软件的配置数据文件。
4. 系统管理服务器解析配置文件内容，将读取的相关配置数据保存在内存中。
5. 系统管理服务器处理注册请求，将保存注册信息，并返回注册响应消息。
6. 系统管理服务器处理配置数据校验请求，核对相关文件的MD5校验码，并根据校验结果，组织并发送配置数据校验响应消息。
7. 如果配置数据校验不通过，向相关的单板系统管理模块发送配置数据更新信息，将系统管理服务器侧的配置数据更新到单板系统管理模块。

单板系统管理模块侧：

1. 单板系统管理模块启动。
2. 单板系统管理模块读取软件管理配置信息，了解本板有哪些业务应用。
3. 单板系统管理模块向系统管理服务器发送注册请求。
4. 在注册成功后，单板系统管理模块读取本板各个应用的配置文件，并计算配置文件的MD5码。
5. 单板系统管理模块针对本板上各业务应用组织配置数据校验请求，发往系统管理服务器。
6. 当单板系统管理模块收到配置数据校验失败响应后，等待并处理系统管理服务器下发的配置更新请求，并更新本地相应的配置数据文件。
7. 单板系统管理模块读取配置数据文件，解析配置数据文件内容，并将配置数据保存在本地内存中。

#### 配置数据获取

##### 业务设计

业务设计如下：

1. 系统管理服务器收到客户端下发的配置数据获取请求。
2. 系统管理服务器从内存保存的配置数据中，查找请求相关数据，组织响应包发送至客户端。

##### 业务流程

图4‑9配置数据获取流程图

业务流程描述：

1. 客户端向系统管理服务器下发配置数据获取请求。
2. 系统管理服务器收到请求后，解析请求消息，分析客户端获取配置数据的过滤条件。
3. 系统管理服务器根据过滤条件，从内存中保存的配置数据中符合的数据记录。
4. 系统管理服务器将查找到符合条件的配置数据记录汇总，进行打包，组织成响应消息，如果配置数据记录很多，则组织多个响应消息。
5. 系统管理服务器将响应消息发送至客户端。
6. 客户端收到响应消息后，解析并获取相关配置数据记录。
7. 客户端将配置数据显示到界面上。

#### 配置数据更新

##### 业务设计

业务设计如下：

1. 系统管理服务器收到客户端下发的配置数据更新请求。
2. 系统管理服务器修改内存数据，并更新本地相关配置文件。
3. 系统管理服务器组织响应消息，发送至客户端。
4. 系统管理服务器下发配置数据更新指令到单板系统管理模块。
5. 单板系统管理模块收到更新指令后，更新相关配置文件。
6. 单板系统管理模块在更新完配置文件后，通知相关业务应用，配置更新。

##### 业务流程



图4‑10配置数据更新流程图

业务流程描述：

1. 客户端向系统管理服务器下发配置数据更新请求。
2. 系统管理服务器收到请求后，解析请求消息，分析客户端更新的配置数据。
3. 系统管理服务器更新内存中，相关配置表的数据。
4. 系统管理服务器根据内存中保存的配置数据，重写配置文件，并进行文件更新。
5. 系统管理服务器组织响应消息，并发送至客户端。
6. 系统管理服务器将配置更新消息转发至单板系统管理模块。
7. 单板系统管理模块收到配置文件更新指令后，更新本地配置文件。
8. 单板系统管理模块组织响应消息，发送给系统管理服务器。
9. 单板系统管理模块组织配置数据更新通知，并发送给相关业务应用。
10. 业务应用软件收到配置数据更新通知后，发送响应消息给单板系统管理模块。

#### 配置数据校验

##### 业务设计

1. 系统管理服务器启动时，计算管辖范围内的所有管理对象的配置文件的校验码，并保存在本地。
2. 单板系统管理模块计算本板各个业务应用的配置数据文件校验码，并保存在本地。
3. 单板系统管理模块针对每一个业务应用，定期组织配置数据校验请求，并发送至系统管理服务器。
4. 系统管理服务器解析校验请求中的校验码，和本地保存的配置文件校验码进行比对，并将校验结果通过校验响应，发送至单板系统管理模块。
5. 单板系统管理模块收到校验响应后，根据校验结果，如果校验结果一致，则退出处理。
6. 如果校验结果不一致，则启动配置数据更新操作，组织配置数据获取请求，发送至系统管理服务器。
7. 系统管理服务器收到配置数据获取请求后，组织相关配置数据，将配置数据内容附加到配置数据获取响应消息中，并返回给单板系统管理模块。
8. 单板系统管理模块收到响应消息后，读取下发的配置数据内容，更新本地配置数据文件，更新本地配置数据校验信息。

##### 业务流程

图4‑11配置数据校验流程图

业务流程描述：

1. 单板系统管理模块定期计算本板上各个业务应用相关配置文件的校验码，并保存在本地。
2. 单板系统管理模块定期针对各个业务应用，以相应配置文件的MD5校验码组织配置数据校验请求，发送至系统管理服务器。
3. 系统管理服务器收到配置数据校验请求后，解析请求内容，获取其中的配置文件校验码。
4. 系统管理服务器将请求中的校验码与本地保存的相应效验码进行比对，查看配置文件内容是否一致。
5. 系统管理服务器组织配置数据校验响应消息，将校验码比对结果发送至单板系统管理模块。
6. 单板系统管理模块收到校验消息后，判断校验结果，如果校验结果一致，则退出配置校验处理。
7. 如果校验结果不一致，单板系统管理模块则组织配置数据获取请求，发送至系统管理服务器，获取最新的相关配置数据。
8. 系统管理服务器收到配置数据请求后，组织响应消息，将相应的配置数据信息更新到单板系统管理模块。
9. 单板系统管理模块收到响应消息后，获取其中的配置数据信息，更新本地配置数据文件。
10. 单板系统管理模块在更新完配置文件后，更新响应的校验信息，并保存在本地，供下次校验使用。

#### 配置数据备份与恢复

##### 业务设计

1. 系统管理平台允许用户对配置数据进行导出操作，以实现配置数据备份功能。
2. 系统管理平台允许用户导入配置数据，以实现配置数据的恢复功能。
3. 系统管理服务器支持用户下发配置数据导出指令，当用户进行导出操作时，系统管理服务器将目标配置文件内容发送给客户端。
4. 客户端收到配置数据文件后，将配置数据保存在本地文件中。
5. 系统管理服务器支持用户下发配置文件导入指令，当用户执行导入操作时，将保存在本地的文件内容发送给系统管理服务器。
6. 系统管理服务器收到文件内容后，更新当前配置文件内容。
7. 系统管理服务器更新配置文件内容后，需通知相关单板系统管理服务器，将数据更新下去。
8. 配置数据导出和导入操作中，配置数据应以XML方式组织。

##### 业务流程



图4‑12配置数据备份与恢复流程图

业务流程描述：

1. 用户下发配置导出指令到系统管理服务器。
2. 系统管理服务器解析指令，判断其合法性，若合法则根据指令组织相关配置数据。
3. 系统管理服务器将组织好的配置数据内容附加到配置导出响应消息中，发送给客户端。
4. 客户端收到配置导出响应后，获取配置数据信息，将其保存在本地文件中。
5. 用户下发配置导入指令到系统管理服务器。
6. 系统管理服务器解析指令，判断其合法性，并将指令中附带的配置数据内容保存在内存中。
7. 系统管理服务器用保持在内存中的配置数据更新相应的当前配置数据文件。
8. 系统管理服务器组织响应消息，并发送给客户端。
9. 系统管理服务器组织配置数据更新指令，将其发送至相应的单板系统管理模块。
10. 单板系统管理模块收到配置数据更新指令后，更新本地相关配置文件，并重新读入内存保存。
11. 单板系统管理模块组织配置数据更新响应，发送回系统管理服务器。
12. 单板系统管理模块组织配置数据更新通知，将其发送至相关业务应用，告知其更新配置。
13. 业务应用组织响应消息，发送给单板系统管理模块。

### 故障管理

#### 故障信息存储

##### 业务设计

1. 系统管理平台采用数据库方式存储故障信息记录。
2. 故障信息的类型分为当前故障和历史故障两种，当前故障表示目前还未恢复的故障信息，历史故障表示已经被恢复的故障信息。
3. 系统管理平台在数据库中建立两个数据表，分别存放当前故障信息和历史故障信息。
4. 故障信息保存包括以下内容：故障信息流水号、故障ID、故障类型、故障上报时间、故障对象类型、故障对象ID、故障参数信息。

##### 存储说明

当前故障信息表和历史故障信息表格式一致，包含：故障信息流水号、故障ID、故障类型、故障上报时间、故障对象类型、故障对象ID、故障参数信息。

故障流水号：用于记录告警上报时序的流水号，可以唯一确定一条故障信息，故障流水号为系统管理平台生成，以保证其不会重复。

故障ID：用于区别具体故障的ID号，该ID可以唯一确定一种故障，故障ID是唯一的，由系统管理平台分配。

故障类型：故障类型分为故障信息与故障恢复信息两种，当故障发生时，应上报故障信息；当故障恢复时，应上报故障恢复信息。

故障上报时间：故障上报时间用于表示故障信息上报的精确时间，为UTC时间格式。

故障对象类型与ID：故障对象类型用于表示上报该故障的业务应用的类型。

故障对象ID：用于表示上报该故障的业务应用的ID号，故障对象类型与故障对象ID组合，可唯一确定故障信息由哪个业务应用上报。

故障参数信息：故障参数信息是由业务应用上报的，用于表示故障具体参数的信息，故障参数信息为字符串格式，由上报故障信息的业务应用组织并上报。

#### 故障上报与恢复

##### 业务设计

1. 系统管理平台与业务应用间通过故障上报与故障恢复接口，来实现故障上报和故障恢复。
2. 单板系统管理模块提供故障信息上报队列，用以缓存业务应用上报的故障信息，保证故障信息上报的时序。
3. 系统管理服务器将收到上报的故障信息保存在当前故障表中，并上报客户端，显示给用户。
4. 系统管理服务器收到故障恢复信息后，匹配对应的故障信息，将其从当前故障表中删除，转移到历史故障信息表中保存，并上报客户端，显示故障恢复。
5. 故障信息和故障恢复信息在系统管理平台的传递过程中，由通讯层传输机制保证故障信息传递的可靠性。

##### 业务流程



图4‑13故障上报与恢复流程图

业务流程描述：

1. 当发送故障时，业务应用通过接口上报故障信息到单板系统管理模块。
2. 单板系统管理模块将其放入故障信息上报队列中，等待上报。
3. 单板系统管理模块将故障上报队列中的故障信息上报到系统管理服务器。
4. 系统管理服务器解析故障信息，获取该故障信息的识别信息，并在当前故障中查询，查看是否有重复的故障信息存在。
5. 如果查询到有相同的故障信息存在，则说明该故障信息为重复故障上报，系统管理服务器将该故障信息丢弃。否则说明该故障信息为正常上报，系统管理服务器将该故障信息保存至当前故障信息表中。
6. 系统管理服务器将故障信息上报至客户端。
7. 客户端将收到的故障信息显示在故障信息界面上，供用户查看。
8. 当故障被恢复时，业务应用通过接口上报故障恢复信息到单板系统管理模块。
9. 单板系统管理模块将其放入故障信息上报队列中，将故障恢复信息上报到系统管理服务器。
10. 系统管理服务器解析故障恢复信息，获取该故障信息的识别信息，并在当前故障中查询，查看是否有能匹配的故障信息存在。
11. 如果查询到有匹配的故障信息存在，则说明该故障恢复信息合法，系统管理服务器将该故障恢复信息以及相应的故障信息转移到历史故障信息表中保存。否则说明该故障恢复信息非法，系统管理服务器将该故障恢复信息丢弃。
12. 系统管理服务器将故障恢复信息上报至客户端。
13. 客户端将对应的故障信息从故障信息界面上删除。

#### 故障信息匹配

##### 业务设计

1. 系统管理平台以故障ID、故障类型、故障对象类型、故障对象ID来作为故障信息匹配的依据。
2. 当有故障信息上报时，系统管理平台根据以上匹配条件，进行重复故障过滤，如果在当前故障信息表中，存在以上信息都一致的故障信息记录，则可确定故障信息重复。
3. 当有故障恢复信息上报后，系统管理平台根据以上匹配条件，进行故障信息匹配，如果在当前故障信息表中，存在以上信息都一致的故障信息记录，则可确定匹配成功。

##### 业务流程



图4‑14故障信息匹配流程图

业务流程描述：

1. 当进行故障信息匹配时，系统管理服务器根据匹配信息，组织SQL查询语句。
2. 系统管理服务器通过数据库存在接口，执行SQL查询语句。
3. 数据库执行SQL查询语句，并返回相应的查询记录结果集。
4. 系统管理服务器根据返回的匹配结果，进行相关业务处理。

#### 重复故障过滤

##### 业务设计

1. 系统管理平台对于收到的每个故障上报消息，进行重复故障过滤处理。
2. 系统管理平台根据上报故障信息的故障ID、故障类型、故障对象类型、故障对象ID进行匹配操作。
3. 当系统管理平台确定为故障信息上报重复时，系统管理平台丢弃该故障信息。

##### 业务流程



图4‑15重复故障过滤流程图

业务流程描述：

1. 系统管理服务器收到上报的故障信息。
2. 系统管理服务器启动故障过滤处理。
3. 系统管理服务器根据故障信息内容组织SQL语句，并调用数据库接口，执行，进行故障匹配操作。
4. 数据库执行SQL查询语句，并返回相应的查询记录结果集。
5. 系统管理服务器根据返回的匹配结果，进行相关处理。、
6. 当系统管理服务器判断故障信息非重复上报，则组织SQL语句，将故障信息存入数据库中的当前故障信息表中。
7. 数据库执行SQL插入语句，并返回相应的执行结果。
8. 当系统管理服务器判断故障信息为重复上报，则将该故障信息丢弃，不做任何处理。

#### 故障信息检索

##### 业务设计

1. 系统管理平台支持用户对保存的故障信息进行查询检索。
2. 系统管理平台支持以故障ID、故障类型、故障时间、故障对象等信息为条件进行检索操作。
3. 系统管理平台根据用户制定的检索条件，组织SQL语句，通过数据库接口，进行查询。
4. 系统管理平台将查询到的故障信息记录组包发送给客户端，如果数据量大于一个数据包长，则分包处理。

##### 业务流程



图4‑16故障信息检索流程图

业务流程描述：

1. 客户端下发故障信息查询请求。
2. 系统管理服务器收到请求后，解析请求消息，获取用户下发的检索条件。
3. 系统管理服务器根据用户下发的故障信息检索条件，组织故障信息查询的SQL语句，并通过数据库操作接口，执行SQL语句，进行故障信息查询操作。
4. 数据库执行SQL查询语句，并返回相应的查询记录结果集。
5. 系统管理服务器根据返回的查询结果，将返回的故障信息记录组包，并发送给客户端，如果数据量较大，一个包返回不完，则进行分包处理。
6. 客户端收到查询结果后，将查询到的故障信息显示给用户。

#### 故障信息备份

##### 业务设计

1. 系统管理平台支持用户对保存的历史故障信息进行导出备份操作。
2. 系统管理平台支持以故障ID、故障类型、故障信息时间段、故障对象等信息为条件进行导出备份操作。
3. 系统管理平台根据用户制定的备份条件，组织SQL语句，通过数据库接口，将符合导出备份条件的故障信息记录返回给客户端。
4. 系统管理平台将需备份的故障信息记录组包发送给客户端，如果数据量大于一个数据包长，则分包处理。
5. 客户端收到故障信息数据后，将其备份到本地文件中。

##### 业务流程



图4‑17故障信息备份流程图

业务流程描述：

1. 客户端下发故障信息备份请求。
2. 系统管理服务器收到请求后，解析请求消息，获取用户下发的故障信息备份条件。
3. 系统管理服务器根据用户下发的故障信息备份条件，组织故障信息备份的查询SQL语句，并通过数据库操作接口，执行SQL语句，进行故障信息查询操作。
4. 数据库执行SQL查询语句，并返回SQL语句查询结果。
5. 系统管理服务器根据查询操作返回的故障信息记录，组织响应包，并将其发送至客户端。
6. 客户端收到故障信息记录后，将其备份到本地文件中。

### 业务管理

#### 业务管理通道

##### 业务设计

1. 系统管理平台支持用户进行业务管理操作。
2. 系统管理平台提供用户进行业务管理的通道，负责业务管理信令的路由和传递。
3. 系统管理平台将用户下发的业务管理指令发送到其目的业务应用，并将业务应用返回的业务管理指令执行结果发送给回用户。

##### 业务流程



图4‑18业务管理流程图

业务流程描述：

1. 客户端下发业务管理指令。
2. 系统管理服务器收到业务管理指令后，解析指令消息，根据指令目的地，将指令发送至相应的单板系统管理模块。
3. 单板系统管理模块将业务管理指令发送至相关业务应用。
4. 业务应用执行业务指令，并将执行结果封装在响应消息中，返回单板系统管理模块。
5. 单板系统管理模块收到响应消息后，将响应消息上报至系统管理服务器。
6. 系统管理服务器收到响应消息后，将响应消息发送给客户端。
7. 客户端收到响应消息后，将业务管理结果显示给用户。

### 系统监控

#### 系统资源监控

##### 业务设计

1. 系统管理平台支持对单板系统资源应用情况进行监控。
2. 单板系统管理模块定时采集所在单板的系统资源使用情况数据（CPU使用率、内存使用率），并保存在内存中。
3. 单板系统管理平台对采集的系统资源使用数据进行分析，并与设置的门限值进行对比，当超过门限值时，单板系统管理模块将上报故障信息，当低于门限值时，单板系统管理模块将上报故障恢复信息。
4. 故障门限值信息，以配置方式保存在单板，由单板系统管理模块读取并应用。

##### 业务流程



图4‑19系统资源监控流程图

业务流程描述：

1. 单板系统管理模块监控定时器超时。
2. 单板系统管理模块启动监控信息采集处理，调用接口获取所在单板的CPU使用率和内存使用率数据，并保存在内存中。
3. 单板系统管理模将监控结果与门限值进行比对，当发现监控结果数据高于门限值时，则组织故障信息，上报系统管理服务器。
4. 系统管理服务器保存故障信息，并将故障信息上报到客户端。
5. 客户端收到故障信息后，将故障信息显示给用户。
6. 单板系统管理模将监控结果与门限值进行比对，当发现监控结果数据低于门限值时，则组织故障恢复信息，上报系统管理服务器。
7. 系统管理服务器恢复相关的故障信息，并将故障恢复信息上报到客户端。
8. 客户端收到故障恢复信息后，并将相应的故障信息恢复。

#### 系统监控门限值配置

##### 业务设计

1. 系统监控所需的故障门限值信息以配置方式保存在单板文件系统中。
2. 故障门限值信息由单板系统管理模块读取并应用。
3. 单板系统管理模块启动时，通过配置功能接口，获取故障门限值配置。

##### 业务流程



图4‑20系统健康门限值设置流程图

业务流程描述：

1. 客户端下发监控门限值配置指令。
2. 系统管理服务器分析配置请求。
3. 系统管理服务器更新内存中的配置信息。
4. 系统管理服务器更新本地配置文件。
5. 系统管理服务器回复响应信息到客户端。
6. 系统管理服务器发送配置更新指令到单板系统管理模块。
7. 单板系统管理模块更新本地配置文件。
8. 单板系统管理模将发送相应消息到系统管理服务器。
9. 单板系统管理模块重读配置文件，并更新内存中保存的监控门限值配置，使新的配置生效。

#### 系统监控结果查询

##### 业务设计

1. 系统管理平台支持用户手动查询系统监控结果。
2. 单板系统管理模块将采集到的监控结果保存在内存中，并定期更新。
3. 当用户发送监控结果查询指令时，需指定查询目标单板，系统管理服务器将指令转发到该单板。
4. 单板系统管理模块接收到指令后，将内存保存的监控结果返回给用户。

##### 业务流程



图4‑21系统监控结果查询流程图

业务流程描述：

1. 客户端下发监控结果查询指令。
2. 系统管理服务器收到指令后，解析指令获取其发送目标后，将指令转发至相关的单板系统管理模块上。
3. 单板系统管理模块收到指令后，将内存中保存的监控结果数据封装入响应包，并发送回系统管理服务器。
4. 系统管理服务器将响应包转发至客户端。
5. 客户端收到响应包后，解析响应包，将其中的监控结果数据显示给用户。

### 信息上报

#### 事件信息的存储

##### 业务设计

1. 系统管理平台采用数据库方式存储事件信息记录。
2. 系统管理平台在数据库中建立事件记录表，用于存放上报的事件信息记录。
3. 事件信息保存包括以下内容：事件信息流水号、事件ID、事件上报时间、事件对象类型、事件对象ID、事件参数信息。

##### 存储说明

事件信息表包含：事件信息流水号、事件ID、事件上报时间、事件对象类型、事件对象ID、事件参数信息。

事件流水号：用于记录事件上报时序的流水号，作为主键，可以唯一确定一条事件信息，事件流水号为系统管理平台生成，以保证其不会重复。

事件ID：用于区别具体事件的ID号，该ID可以唯一确定一种事件，事件ID是唯一的，由系统管理平台分配。

事件上报时间：事件上报时间用于表示事件信息上报的精确时间，为UTC时间格式。

事件对象类型：事件对象类型用于表示上报该事件的业务应用的类型。

事件对象ID：用于表示上报该事件的业务应用的ID号，事件对象类型与事件对象ID组合，可唯一确定事件信息由哪个业务应用上报。

事件参数信息：事件参数信息是由业务应用上报的，用于表示事件具体参数的信息，事件参数信息为字符串格式，由上报事件信息的业务应用组织并上报。

#### 事件信息上报

##### 业务设计

1. 系统管理平台与业务应用间通过事件上报接口，来实现事件信息的上报。
2. 单板系统管理模块提供事件信息上报队列，用以缓存业务应用上报的事件信息，保证事件信息上报的时序。
3. 系统管理服务器将收到上报的事件信息保存在事件信息表中，并上报客户端，显示给用户。
4. 事件信息在系统管理平台的传递过程中，为了保证事件信息传递的可靠性，在每次专递中，必须遵循请求->响应模式，并具备重传机制。

##### 业务流程



图4‑22事件上报流程图

业务流程描述：

1. 当发送事件时，业务应用通过接口上报事件信息到单板系统管理模块。
2. 单板系统管理模块将其放入事件信息上报队列中，等待上报。
3. 单板系统管理模块将事件上报队列中的事件信息上报到系统管理服务器。
4. 系统管理服务器收到后，解析事件信息，并将该事件信息保存至事件信息表中。
5. 系统管理服务器将事件信息上报至客户端。
6. 客户端将收到的事件信息显示在故障信息界面上，供用户查看。

#### 事件信息检索

##### 业务设计

1. 系统管理平台支持用户对保存的事件信息进行查询检索。
2. 系统管理平台支持以事件ID、事件时间、事件对象等信息为条件进行检索操作。
3. 系统管理平台根据用户制定的检索条件，组织SQL语句，通过数据库接口，进行查询。
4. 系统管理平台将查询到的事件信息记录组包发送给客户端，如果数据量大于一个数据包长，则分包处理。

##### 业务流程



图4‑23事件检索流程图

业务流程描述：

1. 客户端下发事件信息查询请求。
2. 系统管理服务器收到请求后，解析请求消息，获取用户下发的检索条件。
3. 系统管理服务器根据用户下发的事件信息检索条件，组织事件信息查询的SQL语句，并通过数据库操作接口，执行SQL语句，进行事件信息查询操作。
4. 数据库执行SQL查询语句，并返回相应的查询记录结果集。
5. 系统管理服务器根据返回的查询结果，将返回的事件信息记录组包，并发送给客户端，如果数据量较大，一个包返回不完，则进行分包处理。
6. 客户端收到查询结果后，将查询到的事件信息显示给用户。

#### 事件信息备份

##### 业务设计

1. 系统管理平台支持用户对保存的事件信息进行导出备份操作。
2. 系统管理平台支持以事件ID、事件信息时间段、事件对象等信息为条件进行导出备份操作。
3. 系统管理平台根据用户制定的备份条件，组织SQL语句，通过数据库接口，将符合导出备份条件的事件信息记录返回给客户端。
4. 系统管理平台将需备份的事件信息记录组包发送给客户端，如果数据量大于一个数据包长，则分包处理。
5. 客户端收到事件信息数据后，将其备份到本地文件中。

##### 业务流程



图4‑24事件备份流程图

业务流程描述：

1. 客户端下发事件信息备份请求。
2. 系统管理服务器收到请求后，解析请求消息，获取用户下发的事件信息备份条件。
3. 系统管理服务器根据用户下发的事件信息备份条件，组织事件信息备份的查询SQL语句，并通过数据库操作接口，执行SQL语句，进行事件信息查询操作。
4. 数据库执行SQL查询语句，并返回SQL语句查询结果。
5. 系统管理服务器根据查询操作返回的事件信息记录，组织响应包，并将其发送至客户端。
6. 客户端收到事件信息记录后，将其备份到本地文件中。

### 软件管理

#### 系统管理平台自身管理

##### 业务设计

1. 系统管理平台作为其他软件运行的基础，必须保证自身运行的可靠性。
2. 系统管理平台在运行中发生故障，导致无法继续运行的情况下，应具备恢复运行的能力。
3. 由于系统管理平台是其他业务软件运行的基础，当系统管理平台运行故障时，其他业务软件没有继续运行的意义，因此采用单板重启的方式恢复系统管理平台的运行。
4. 系统管理平台与软件看门狗配合，以实现系统管理平台允许的可靠性。系统管理平台启动后定时“喂狗”，当超过一定时间没有“喂狗”，则软件看门狗可以认为系统管理平台运行异常，重启单板以恢复系统管理平台的运行。

##### 业务流程



图4‑25系统管理平台管理流程图

业务流程描述：

1. 软件看门狗启动，设置并启动单板复位定时器，等待系统管理平台“喂狗”。
2. 系统管理平台启动，设置并启动“喂狗”循环定时器。
3. 系统管理平台“喂狗”定时器超时后，向看门狗发送消息，进行“喂狗”操作。
4. 软件看门狗收到“喂狗”消息后，判断系统管理平台运行正常，复位单板复位定时器。
5. 当软件看门狗单板复位定时器超时，仍未收到“喂狗”消息，则判断系统管理平台运行故障。
6. 软件看门狗复位单板，恢复系统管理平台的运行。

#### 软件管理策略数据存储

##### 业务设计

1. 系统管理平台将软件管理配置信息以配置数据形式保存在本地配置文件中。
2. 系统管理平台启动时，读取相应的配置文件，以获取软件管理配置信息。
3. 系统管理配置信息包括：软件管理信息、软件管理策略。
4. 软件管理信息包括：被管理的软件、软件存放路径、软件启动顺序；通过软件管理信息，系统管理平台可以了解系统中都有哪些软件需要被管理，到哪里能够找到这些软件，以及软件启动的顺序。
5. 软件管理策略包括：软件启动管理策略、软件运行管理策略、软件退出策略；通过软件管理策略，系统管理平台能了解到在软件启动到退出的过程中，采用什么样的方式对软件进行管理。

##### 数据存储

软件管理信息定义：

struct Soft\_Mng\_Info

{

UINT8 nIndex; //应用软件索引号，同时代表启动顺序

UINT8 nSoftType； //应用软件类型

UINT8 nSoftId； //应用软件ID

char szSoftName[20]; //应用软件名称

char szSoftPath[MAX\_PATH]; //应用软件可执行文件路径

};

struct Soft\_Mng\_ Policy

{

UINT8 nStartPolicy; //软件启动处理策略

//0：顺序启动 启动成功后再启动下一个

//1：循环启动 启动失败后先启动后续软件

UINT8 nStartTimerLimit; //软件启动等待响应时长，单位秒

UINT8 nStartExceptionPolicy; //软件启动异常处理策略

//0重启软件 1重启单板

UINT8 nRunExceptionPolicy; //软件运行异常管理策略

//0 恢复软件运行 1报错并等待人为干预

UINT8 nRunRestoreMode; //软件恢复运行处理方式

//0通知进程退出后重启 1杀死进程后重启 2重启单板

UINT8 nStopPolicy; //软件退出处理策略

//0通知进程退出 1直接杀死进程

UINT8 nStopTimerLimit; //软件退出等待时长，单位秒

UINT8 nStopTimeoutPolicy; //软件退出超时处理策略

//0通知进程退出 1直接杀死进程

};

XML方式请参考4.2.1.1配置数据存储章节

#### 软件管理策略配置

##### 业务设计

1. 系统管理平台允许用户对软件管理策略进行配置。
2. 用户通过客户端下发软件管理策略配置指令，系统管理平台将配置数据保存在本地配置文件中。
3. 单板系统管理模块读取新的软件管理配置策略数据，并使这些策略生效。

##### 业务流程



图4‑26软件管理策略配置流程图

业务流程描述：

1. 客户端向系统管理服务器下发软件管理策略数据更新请求。
2. 系统管理服务器收到请求后，根据请求消息的中的内容，更新本地配置文件。
3. 系统管理服务器组织响应消息，并发送至客户端。
4. 系统管理服务器将更新的配置文件内容下发至单板系统管理模块。
5. 单板系统管理模块收到配置文件更新消息后，更新本地配置文件内容。
6. 单板系统管理模块组织响应消息，发送给系统管理服务器。
7. 单板系统管理模块重读软件管理策略数据，并使之生效。

#### 软件启动管理

##### 业务设计

1. 系统管理平台根据软件管理配置，启动本单板的业务软件。
2. 系统管理平台根据软件管理配置中设定的启动顺序，以及软件存依次启动各个业务管理软件。
3. 系统管理平台根据软件管理配置中的软件存放路径，找到业务软件的可执行文件，启动该业务软件。
4. 系统管理平台根据软件管理配置中的软件启动策略，对软件启动过程中遇到的问题进行相关处理。

##### 业务流程



图4‑27软件启动管理流程图

业务流程描述：

1. 系统管理平台开始软件启动流程。
2. 系统管理平台读取软件管理配置信息。
3. 系统管理平台案中案软件启动顺序将软件管理信息进行排序。
4. 系统管理平台根据软件启动顺序，查找未启动的业务软件。
5. 如果没找到未启动的业务软件，则软件启动结束，退出处理。
6. 否则根据软件管理配置信息，查找到该软件保存路径下的业务软件可执行文件。
7. 调用系统接口，启动业务应用软件。
8. 启动定时器，等待业务软件启动成功响应。
9. 如果收到响应消息，说明业务软件成功，则停止定时器，返回步骤4继续查找未启动的业务软件。
10. 如果定时器超时，则说明业务软件启动异常。
11. 当业务软件启动异常，系统管理平台判断软件启动顺序策略配置，如果配置为1（循环启动），则跳到步骤4，继续启动后续业务软件。
12. 如果软件启动顺序策略配置为0（顺序启动），则系统管理平台继续判断软件启动异常策略配置，如果配置为0（重启软件），则跳到步骤7，继续启动业务软件，如果为1（重启单板）则系统管理平台重启单板，重新开始软件启动流程。

#### 软件运行监控

##### 业务设计

1. 系统管理平台支持对应用软件运行情况进行监控。
2. 系统管理平台对软件运行状态进行被动式监控，系统管理平台为每个纳入软件管理的业务软件建立健康性检查控制块，并设定监控周期，等待业务软件的健康性报告。
3. 业务软件在运行过程中定期向系统管理平台发送健康性报告，如果若干周期内，业务软件没有上报健康性报告，则系统管理平台认为该业务软件运行异常，按照软件异运行异常处理策略进行相关操作。

##### 业务流程



图4‑28软件运行监控流程图

业务流程描述：

1. 单板系统管理模块建立软件监控控制块，为每个业务软件分配监控计数器。
2. 单板系统管理模块申请监控循环定时器，并启动。
3. 业务软件启动后，定期向单板系统管理模块发送健康性报告。
4. 单板系统管理模块收到定时器超时消息后，将软件监控计数器加1。
5. 单板系统管理模块收到业务软件发送的健康性报告后，将对应的软件监控计数器置0。
6. 当业务软件运行异常时，停止向单板系统管理模块发送健康性报告。
7. 单板系统管理模块判断软件监控计数器达到一定数值时，判断相应的业务软件运行异常，按照软件运行异常处理处理进行相关处理。

#### 软件异常管理

##### 业务设计

1. 系统管理平台支持对运行异常的应用软件进行管理
2. 系统管理平台判断应用软件运行异常后，根据软件运行异常处理策略进行相关处理。

##### 业务流程



图4‑29软件异常管理流程图

业务流程描述：

1. 单板系统管理模块判定业务软件运行异常。
2. 单板系统管理模块组织故障信息，并上报系统管理服务器。
3. 系统管理服务器收到故障信息后，进行相关处理，并返回响应消息。
4. 系统管理服务器将故障消息上报客户端。
5. 客户端收到故障信息后，返回响应消息，并将故障信息显示给用户。
6. 如果系统管理平台的软件运行异常处理策略配置为“重启软件”，则系统管理平台重启业务软件，当当软件恢复运行后，跳至步骤12。如果系统管理平台的软件运行异常处理策略配置为“用户干预”，则单板系统管理模块等待用户下发指令。
7. 用户看到软件运行故障信息后，下发软件启动指令到系统管理服务器。
8. 系统管理服务器收到指令后，将指令下发至相应的单板系统管理模块。
9. 单板系统管理模块收到软件启动指令后，启动业务软件。
10. 当业务软件启动成功后，单板系统管理模块组织响应消息，发送回系统管理服务器。
11. 系统管理服务器将响应消息发送至客户端，通知用户软件启动成功。
12. 当业务软件运行正常后，单板系统管理模块将组织故障恢复信息，并上报系统管理服务器。
13. 系统管理服务器收到故障恢复信息后，进行相关处理，并返回响应消息。
14. 系统管理服务器将故障恢复消息上报客户端。
15. 客户端收到故障恢复信息后，返回响应消息，并将相关故障恢复。

#### 软件退出管理

##### 业务设计

1. 系统管理平台支持对运行的应用软件进行退出处理。
2. 当系统管理平台需要业务软件退出运行时，根据软件退出处理策略进行相关处理。

##### 业务流程



图4‑30软件退出管理流程图

业务流程描述：

1. 单板系统管理模块执行软件退出处理。
2. 如果软件退出策略为“杀死进程”，则跳至步骤6；否则单板系统管理模块组织软件退出通知信息，发送至业务软件。
3. 单板系统管理模块启动软件停止定时器。
4. 业务软件收到退出通知后，是否相关资源，返回响应消息后，退出运行。。
5. 单板系统管理模块收到软件退出响应消息后，停止定时器。如果定时器超时仍未收到软件退出响应消息。则跳至步骤6。
6. 单板系统管理模块调用系统接口，杀死业务软件进程。

#### 软件运行控制

##### 业务设计

1. 系统管理平台支持用户对应用软件的运行进行控制。
2. 系统管理平台支持用户下发针对业务软件的软件启动与软件退出指令。

##### 业务流程

图4‑31软件运行控制流程图

业务流程描述：

1. 用户通过客户端下发软件退出指令。
2. 系统管理服务器收到指令后，转发给相关的单板系统管理模块。
3. 单板系统管理模块收到软件退出指令后，按照软件退出处理流程，停止软件运行。
4. 单板系统管理结束软件运行后，组织响应消息，返回系统管理服务器。
5. 系统管理服务器收到响应消息后，将其上报给客户端，表示处理完成。
6. 用户通过客户端下发软件启动指令。
7. 系统管理服务器收到指令后，转发给相关的单板系统管理模块。
8. 单板系统管理模块收到软件启动指令后，按照软件启动处理流程，启动业务软件。
9. 单板系统管理成功启动业务软件后，组织响应消息，返回系统管理服务器。
10. 系统管理服务器收到响应消息后，将其上报给客户端，表示处理完成。

#### 软件更新管理

##### 业务设计

1. 系统管理平台支持通过与客户端经过相关交互，实现业务软件的更新功能。
2. 业务软件更新功能分为以下几个步骤：软件验证、软件更新、软件启动、软件回滚。
3. 当需进行软件更新时，客户端首先需要对软件版本进行验证，查看软件版本是否正确。软件版本验证信息，客户端以配置信息的方式从系统管理服务器获取。
4. 在通过软件验证后，客户端通过FTP文件传输方式，将软件文件更新到单板系统管理模块。
5. 在软件更新完成后，单板系统管理模块启动业务软件。如果软件启动成功，则软件更新流程结束。否则，单板系统管理模块将上报故障信息，并阻止该软件启动。
6. 当客户端收到软件更新故障后，需由用户介入，进行软件回滚。通过更新正确的软件版本，恢复软件运行。

##### 业务流程



图4‑32软件更新流程图

业务流程描述：

1. 客户端启动时，向服务器发起配置数据获取请求，获取版本匹配信息，以备后续使用。
2. 系统管理服务器收到软件版本配置信息请求后，返回版本配置信息。
3. 当用户执行软件更新时，客户端在软件更新前，首先对软件版本进行验证，查看软件版本是否合法。
4. 当软件版本通过验证后，客户端发送文件传输指令，更新软件相关文件。通过FTP文件传输流程，实现文件的更新。
5. 单板系统管理模块进行文件传输协商，并在本地创建文件。
6. 单板系统管理模块将传输的文件内容写入本地文件。
7. 单板系统管理模块在文件写操作完成后，关闭本地文件，并结束文件传输流程。
8. 单板系统管理模块在文件更新完毕后，执行软件退出处理，停止业务软件的运行。
9. 单板系统管理模块进行文件切换，将软件相关文件切换为新的版本。
10. 单板系统管理模块切换文件完成后，执行软件启动处理，启动业务软件。
11. 如果业务软件启动失败，单板系统管理模块上报故障信息
12. 单板系统管理模块阻止软件启动，并等待用户进行处理
13. 客户端收到故障信息后，显示给用户，并提示用户进行软件回滚。

### 系统安全

#### 安全信息存储

##### 业务设计

1. 系统管理平台以配置数据形式保存安全信息用户权限相关的各种信息。
2. 系统管理服务器启动时，读取安全信息配置，将其保存在内存中，供用户权鉴使用。
3. 系统管理平台允许拥有权限的用户，对用户权限信息进行配置。
4. 当用户更新安全信息配置，系统管理服务器需要将更新后的相关配置文件同步到其他系统管理服务器。

##### 数据说明

1. 系统管理平台以用户信息、用户组信息、用户组权限信息组合，来实现系统安全信息的保存。
2. 用户信息包括用户ID、所属的用户组ID、用户标识、用户密码、用户名、用户描述等。（其中用户密码非明文，以MD5码形式保存）
3. 用户组信息包括用户组ID、用户组名称、用户组描述。
4. 用户组权限包括用户组ID以及其具备的权限信息。

##### 业务流程

安全信息的初始化、更新、删除等相关业务流程请参考4.2.1数据配置章节

#### 用户权鉴处理

##### 业务设计

1. 系统管理平台在用户登录时，对用户进行权鉴处理。
2. 客户端发送登录请求中，需附带必要的用户信息，作为权鉴处理的依据。
3. 系统管理服务器收到登录请求后，根据其中附带的用户信息，进行权鉴处理，判断用户信息是否合法。
4. 当用户信息没有通过权鉴处理时，系统管理服务器认为用户非法，拒绝用户登录系统。
5. 当用户信息通过权鉴处理时，系统管理服务器认为用户合法，在响应消息中，附带该用户的权限信息。

##### 业务流程



图4‑33用户权鉴流程图

业务流程描述：

1. 客户端发送登录请求，并在请求中附带用户信息。
2. 系统管理服务器收到请求后，解析请求消息，获取请求中的用户信息。
3. 系统管理服务器将请求中的用户信息（用户标识和用户密码）和安全配置中的用户信息进行核对。
4. 如果核对不符，则系统管理服务器认为用户非法，返回登录失败响应，拒绝用户登录。
5. 如果核对一致，则系统管理服务器从配置信息中获取该用户权限信息，并分配本次登录的安全凭证，将其附加到相应消息中发送给客户端，允许用户登录。

#### 信令安全保证

##### 业务设计

1. 系统管理平台以安全凭证来保证在信令处理上的安全性。
2. 当客户端登录请求通过系统管理服务器的权鉴检查后，系统管理平台为该客户端分配安全凭证，并通过响应消息返回给客户端。
3. 客户端在后续下发的任何指令中，必须附带该安全凭证。
4. 当系统管理服务器收到指令中无安全凭证，则拒绝该指令。
5. 系统管理服务器分配的安全凭证有效期从客户端登录到该客户度退出登录结束。
6. 安全凭证格式为一个4字节随机数，由系统管理服务器计算分配

##### 业务流程

业务流程请参见章节4.2.7.2.2。

### 数据同步

#### 管理范围内数据同步

##### 业务设计

1. 在同一管理范围内，为保证系统管理平台中各个系统管理服务器对外提供一致的系统管理服务，因此必须保证相关数据一致性。
2. 同一管理范围内的系统管理服务器间，通过数据同步来保证相关数据一致性。
3. 数据同步请求由一方系统管理服务器发起，另一方系统管理服务器接收并进行同步处理。
4. 发送的数据同步请求中，附带需同步的数据信息。如果需同步的数据超过一个数据包长度，则进行分包处理。
5. 接收方收到数据同步请求后，进行数据同步处理完成后，需返回数据同步响应。
6. 需在同一管理范围内进行数据同步的数据类别如下：软件管理相关配置数据、软件业务配置数据、安全信息数据、系统信息数据。

##### 业务流程



图4‑34管理范围内数据同步流程图

业务流程描述：

1. 当需要进行数据同步时，系统管理服务器A向系统管理服务器B发送数据同步请求。
2. 系统管理服务器B收到同步请求后，将请求数据包中的同步数据取出，保存在内存中。
3. 当同步数据收完后，系统管理服务器B将保持在内存中的数据，写入相应的本地文件中。
4. 系统管理服务器B重读本地文件，使同步过来的数据生效。
5. 系统管理服务器B组织响应消息，返回系统管理服务器A，表示数据同步完成。

#### 管理范围间的数据同步

##### 业务设计

1. 在不同管理范围之间，为保证系统管理平台中各个系统管理服务器都能对客户端提供相同的接入服务，以及系统管理信令能够在系统管理平台中进行正确的路由，须保证相关数据一致性。
2. 不同管理范围的系统管理服务器之间，通过数据同步来保证相关数据的一致性。
3. 管理范围间的数据同步请求由一方系统管理服务器发起，其他所有系统管理服务器接收并进行同步处理。
4. 发送的数据同步请求中，附带需同步的数据信息。如果需同步的数据超过一个数据包长度，则进行分包处理。
5. 接收方收到数据同步请求后，进行数据同步处理完成后，需返回数据同步响应。
6. 需在管理范围间进行数据同步的数据类别如下：安全信息数据、系统信息数据。

##### 业务流程



图4‑35管理范围间数据同步流程图

业务流程描述：

1. 当需要进行数据同步时，系统管理服务器A向其他管理范围的系统管理服务器发送数据同步请求。
2. 系统管理服务器B收到同步请求后，将请求数据包中的同步数据取出，保存在内存中。
3. 当同步数据收完后，系统管理服务器B将保持在内存中的数据，写入相应的本地文件中。
4. 系统管理服务器B重读本地文件，使同步过来的数据生效。
5. 系统管理服务器B组织响应消息，返回系统管理服务器A，表示数据同步完成。
6. 系统管理服务器C收到同步请求后，重复步骤2~5完成本地的数据同步处理。
7. 系统管理服务器A收到所有系统管理服务器的响应消息后，退出数据同步处理。

### 信令传递与转发

#### 业务注册

##### 业务设计

1. 为了能使系统管理服务器能够正确的进行下行的信息传递，单板系统管理模块启动后，需向系统管理服务器进行注册。
2. 单板系统管理模块向所属的系统管理服务器发送注册请求，注册请求中需包含本单板内所有管理范围内的业务软件信息
3. 系统管理服务器收到注册请求后，将业务软件信息取出，保存在内存中，供信息路由使用。

##### 业务流程



图4‑36业务注册流程图

业务流程描述：

1. 单板系统管理模块启动后，向系统管理服务器发送注册请求。
2. 系统管理服务器收到注册请求后，将注册信息保存在内存中，为信息路由做准备。
3. 系统管理服务器发送注册响应信息给单板系统管理模块。

#### 系统信息

##### 业务设计

1. 为实现管理范围间的信息转发，系统管理平台需要保存产品系统信息。
2. 系统管理平台通过系统信息了解产品构成，产品系统信息包括：包含管理范围的划分、每个管理范围的系统管理服务器通讯信息、每个管理范围包含哪些业务应用等。
3. 产品系统信息以文件形式保存，由系统管理服务器读取并使用。
4. 系统管理平台需保证产品系统信息的一致性，当由于用户修改配置导致产品系统信息发生变化时，系统管理平台需要通过数据同步机制，将修改后的产品系统信息同步到系统管理平台内的所有系统管理服务器。

#### 客户端登录信息

##### 业务设计

1. 为实现管理范围间的上行信令传递，系统管理平台需要保存并实时更新客户端登录信息。
2. 系统管理平台通过客户端登录信息了解都有哪些客户端登录到系统管理平台，各自的登录点在哪里。
3. 客户端登录信息保存在每个系统管理服务器的内存中。
4. 系统管理平台需保证客户端登录信息的一致性，当客户端登录到一个系统管理服务器，或从系统管理服务器登出时，该系统管理服务器都需要将该客户端登录情况告知其他系统管理服务器。

#### 信令转发

##### 业务设计

1. 系统管理平台通过系统管理信令中的发送源于发送目的，来确保系统管理信令的正确传递。
2. 系统管理信令中包含信令的发送源与发送目的。
3. 当信令发送目的为同一管理范围内的对象时，系统管理服务器通过注册信息查询，找到信令目的对应的单板系统管理模块，将信令先发送至单板系统管理模块。再由单板系统管理模块发送至业务应用。
4. 当信令发送目的非同一管理范围内的对象时，系统管理服务器通过产品系统信息，找到信令目的对应的系统管理服务器，将信令发送至目的系统管理服务器，在由其进行后续转发。

##### 管理范围内信令路由业务流程



图4‑37同一管理范围内信令传递流程图

业务流程描述：

1. 客户端下发系统管理指令。
2. 系统管理服务器收到指令后，判断指令目标是否为本管理范围。
3. 如果目标位于同一管理范围，则遍历注册信息，查找到目标所在的单板系统管理模块。
4. 系统管理服务器将指令发送至目标单板系统管理模块，并记录指令的源地址。
5. 单板系统管理模块收到指令后，根据指令目标，将指令发送给相应的业务应用。
6. 业务应用执行处理后返回响应消息，响应消息中，反填原指令中的目的与源信息。
7. 单板系统管理模块收到响应消息后，将响应消息上报系统管理服务器
8. 系统管理服务器收到响应消息后，根据指令的源地址，将响应消息返回客户端。

##### 跨管理范围信令路由业务流程



图4‑38跨管理范围信令路由流程图

业务流程描述：

1. 客户端下发系统管理指令（客户端登录系统管理服务器A,指令目标为系统管理服务器B）。
2. 系统管理服务器A收到指令后，判断指令目标非本管理范围内对象，查找产品系统信息，找到目标对应的系统管理服务器B。
3. 系统管理服务器A保存指令的源地址。
4. 系统管理服务器A修改指令源地址为本服务器地址后，将指令发送至系统管理服务器B。
5. 系统管理服务器B收到指令后，判断指令目标为本管理范围内对象，遍历注册信息，查找到目标所在的单板系统管理模块。
6. 系统管理服务器B保存指令的源地址。
7. 系统管理服务器B修改指令源地址为本服务器地址后,将指令发送至目标单板系统管理模块。
8. 单板系统管理模块收到指令后，根据指令目标，将指令发送给相应的业务应用。
9. 业务应用执行处理后返回响应消息，响应消息中，反填原指令中的目的与源信息。
10. 单板系统管理模块收到响应消息后，将响应消息上报系统管理服务器。
11. 系统管理服务器B收到响应消息后，根据记录的指令源地址，修改响应消息的目标地址。
12. 系统管理服务器B将响应消息发送到系统管理服务器A。
13. 系统管理服务器A收到响应消息后，根据记录的指令源地址，修改响应消息的目标地址。
14. 系统管理服务器A将响应消息发送到客户端。

##### 信令上报业务流程



图4‑39信令上报流程图

业务流程描述：

1. 业务应用上报事件信息。
2. 单板系统管理模块收到事件信息后，发送响应消息，表示已收到该事件。
3. 单板系统管理模块将事件信息上报所属的系统管理服务器B。
4. 系统管理服务器B收到事件信息后，发送响应消息，表示已经收到该事件。
5. 系统管理服务器B保存事件信息，将其存入数据库。
6. 系统管理服务器B将事件信息上报给登录服务器B的客户端B。
7. 客户端B收到事件信息后，返回响应信息，并将事件信息显示给用户。
8. 系统管理服务器B查询产品系统信息，查找需要进行通知的其他系统管理服务器。
9. 系统管理服务器B，将事件信息通知给系统管理服务器A。
10. 系统管理服务器A保存事件信息，将其存入数据库。
11. 系统管理服务器A将事件信息上报给登录服务器B的客户端A。
12. 客户端A收到事件信息后，返回响应信息，并将事件信息显示给用户。

## 接口设计

### 接口标识和图表



图4‑40系统管理平台接口图

| **接口名称** | **接口类型** | **接口描述** |
| --- | --- | --- |
| E-IF1 | 外部接口 | 与客户端进行系统管理信令交互接口 |
| E-IF2 | 外部接口 | 与业务应用进行系统管理信令交互接口 |
| I-IF1 | 内部接口 | 系统管理服务器与单板系统管理模块信令交互接口 |
| I-IF2 | 内部接口 | 系统管理服务器间信令交互接口 |

### 外部接口描述

#### E-IF1：系统管理平台与客户端信令交互接口

本接口描述文档，请查阅文档《系统管理平台-接口设计说明书》中，关于客户端与系统管理服务器通讯章节。

#### E-IF2：系统管理平台与业务应用信令交互接口

本接口描述文档，请查阅文档《系统管理平台-接口设计说明书》中，关于单板系统管理模块与业务软件通讯章节。

### 内部接口描述

#### I-IF1：系统管理服务器与单板系统管理模块信令交互接口

本接口描述文档，请查阅文档《系统管理平台-接口设计说明书》中，关于系统管理服务器与单板系统管理模块通讯章节。

#### I-IF2：系统管理服务器间信令交互接口

本接口描述文档，请查阅文档《系统管理平台-接口设计说明书》中，关于系统管理服务器间通讯章节。

# 需求的可追踪性

## 部署需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-ARRAY-1 | 支持单域单网元部署 | 4.3.9.1  4.3.9.3 |
| REQ-ARRAY-2 | 支持单域多网元部署 | 4.3.9.2  4.3.9.3 |
| REQ-ARRAY-3 | 系统管理信令的路由 | 4.3.9.3 |
| REQ-ARRAY-4 | 系统管理平台保证系统管理信息在平台内传输的可靠性 | 4.3.9 |
| REQ-ARRAY-5 | 系统管理平台内的系统信息同步 | 4.3.8.2 |
| REQ-ARRAY-6 | 系统管理平台支持多客户端登录 | 4.3.7.2 |
| REQ-ARRAY-7 | 系统管理平台支持客户端同时多点连接 | 4.3.7.2  4.3.8  4.3.9.3 |
| REQ-ARRAY-8 | 系统管理平台内安全信息同步 | 4.3.8 |

## 数据配置功能需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-CONFIG-1 | 系统管理平台对外提供统一的配置数据访问与操作功能 | 4.3.1.1  4.3.1.2  4.3.1.3 |
| REQ-CONFIG-2 | 统一的系统管理配置处理逻辑，不受新增配置业务影响 | 4.2.1 |
| REQ-CONFIG-3 | 系统管理平台支持不同数据格式配置需求 | 4.2.1 |
| REQ-CONFIG-4 | 系统管理平台与客户端之间配置数据交互 | 4.3.1.2  4.3.1.3 |
| REQ-CONFIG-5 | 系统管理平台与业务应用之间配置数据交互 | 4.3.1.1  4.3.1.3 |
| REQ-CONFIG-6 | 系统管理平台支持XML数据解析 | 4.2.1  4.3.1 |
| REQ-CONFIG-7 | 系统管理平台具备配置数据存储功能 | 4.2.1  4.3.1.1 |
| REQ-CONFIG-8 | 系统管理平台支持业务配置数据同步 | 4.3.8.1 |
| REQ-CONFIG-9 | 系统管理平台支持配置数据校验 | 4.3.1.4 |
| REQ-CONFIG-10 | 系统管理平台支持配置数据回滚功能 | 4.3.1.5 |
| REQ-CONFIG-11 | 配置数据备份与恢复 | 4.3.1.5 |

## 故障管理功能需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-ALARM-1 | 故障的上报通道 | 4.3.2.2 |
| REQ-ALARM-2 | 故障信息定位功能 | 4.3.2.1 |
| REQ-ALARM-3 | 故障信息识别功能 | 4.3.2.3  4.3.2.4 |
| REQ-ALARM-4 | 故障信息保存及维护 | 4.3.2.1 |
| REQ-ALARM-5 | 故障信息检索 | 4.3.2.5 |
| REQ-ALARM-6 | 故障信息备份功能 | 4.3.2.6 |

## 业务管理功能需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-SERVICE-1 | 系统管理平台提供业务管理通道 | 4.3.3 |

## 系统监控功能需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-MONITOR-1 | 系统管理平台支持对单板与进程进行监控 | 4.3.4.1 |
| REQ-MONITOR-2 | 系统管理平台监控功能上报故障信息 | 4.3.4.1 |
| REQ-MONITOR-3 | 系统管理平台支持对监控门限值进行配置 | 4.3.4.2 |
| REQ-MONITOR-4 | 系统管理平台支持对监控结果进行查询 | 4.3.4.3 |

## 系统信息上报功能需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-EVENT-1 | 事件信息上报 | 4.3.5.2 |
| REQ-EVENT-2 | 事件信息的保存与维护 | 4.3.5.1 |
| REQ-EVENT-3 | 事件信息检索 | 4.3.5.3 |
| REQ-EVENT-4 | 事件信息备份功能 | 4.3.5.4 |

## 软件管理功能需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-SOFT-1 | 系统管理平台软件管理功能 | 4.3.6 |
| REQ-SOFT-2 | 系统管理平台自身软件故障处理 | 4.3.6.1 |
| REQ-SOFT-3 | 系统管理平台业务软件的启动管理 | 4.3.6.4 |
| REQ-SOFT-4 | 系统管理平台业务软件启动过程管理 | 4.3.6.4 |
| REQ-SOFT-5 | 系统管理平台监控业务软件运行 | 4.3.6.5 |
| REQ-SOFT-6 | 系统管理平台处理业务软件运行异常 | 4.3.6.6 |
| REQ-SOFT-7 | 系统管理平台支持软件管理策略配置 | 4.3.6.2  4.3.6.3 |
| REQ-SOFT-8 | 系统管理平台支持软件结束通知 | 4.3.6.7 |
| REQ-SOFT-9 | 系统管理平台支持文件下载机制 | 4.3.6.9 |
| REQ-SOFT-10 | 系统管理平台支持业务软件的更新 | 4.3.6.9 |
| REQ-SOFT-11 | 系统管理平台支持客户端对业务软件运行控制功能 | 4.3.6.8 |

## 系统安全功能需求

| **需求 ID** | **需求标题** | **对应章节** |
| --- | --- | --- |
| REQ-SAFE-1 | 系统管理平台提供对用户权鉴机制 | 4.3.7.2 |
| REQ-SAFE-2 | 系统管理平台支持灵活的用户权鉴方式 | 4.3.7.2 |
| REQ-SAFE-3 | 系统管理平台灵活的用户权限设定 | 4.3.7.1 |
| REQ-SAFE-4 | 系统管理平台的用户权限告知 | 4.3.7.2 |
| REQ-SAFE-5 | 客户端登录后的信令安全保证 | 4.3.7.3 |
| REQ-SAFE-6 | 用户操作日志自动记录 | 4.3.7.4 |
| REQ-SAFE-7 | 系统管理平台支持安全配置 | 4.3.7.1  4.3.1.3 |

# 注释

SMP： System Manage Platfram 系统管理平台