ICS 35.020
L 09
备案号:36764-2013

DB11

北　京　市　地　方　标　准

DB11/T 948.4—2013

电梯运行安全监测信息管理系统技术规范 第4部分　采集设备和平台的通信协议与数据格式

Technical specifications for lifts, escalators and moving walks operation safety monitoring information management system

Part 4: Communication protocol and data format of acquisition equipment and platform

2013 - 01 - 31 发布　　2013 - 05 - 01 实施

北京市质量技术监督局　发布

目　　次

前　言

DB11/T 948《电梯运行安全监测信息管理系统技术规范》分为以下几个部分：

—— 第 1 部分：系统总体结构；

—— 第 2 部分：电梯基础信息与数据格式；

—— 第 3 部分：采集设备编码规则；

—— 第 4 部分：采集设备和平台的通信协议与数据格式；

—— 第 5 部分：传输网络要求；

—— 第 6 部分：监测数据存储要求；

—— 第 7 部分：图像子系统技术要求；

—— 第 8 部分：采集设备技术要求；

—— 第 9 部分：电梯运行数据格式与输出要求；

—— 第 10 部分：采集设备安装验收规范；

—— 第 11 部分：平台技术要求；

—— 第 12 部分：系统信息安全规范；

—— 第 13 部分：平台维护要求。

本部分为 DB11/T 948 的第 4 部分。

本部分由北京市质量技术监督局提出。

本部分由北京市质量技术监督局归口。

本部分由北京市质量技术监督局组织实施。

本部分主要起草单位：北京市质量技术监督局、北京蓝光宏达科技有限公司、北京市标准化研究所、沈阳蓝光网络数据技术有限公司。

本部分主要起草人：周建民、王宏剑、郑刚、宋新军、李亮华、宋国建、邢磊、杨毅、李勇、陈凌、陈辉。

电梯运行安全监测信息管理系统技术规范
第 4 部分 采集设备和平台的通信协议与数据格式

1 范围

本部分规定了电梯采集设备和电梯运行安全监测信息管理平台（以下简称“平台”）之间的通信协议与数据格式。

本部分适用于电梯采集设备和平台之间的通信和数据传输。

2 规范性引用文件

下列文件对于本文件的应用是必不可少。凡是注日期的引用文件，仅注日期的版本适用本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 24475—2009 电梯远程报警系统

GB/T 24476—2009 电梯、自动扶梯和自动人行道数据监视和记录规范

3 术语和定义

GB/T 24475—2009、GB/T 24476—2009中确定的以及下列术语和定义适用于本部分。

3.1

电梯运行安全监测信息管理平台 lifts, escalators and moving walks operation safety monitoring information management platform

具备电梯管理、故障报警、远程监测分析、数据存储等功能的集中式系统平台。

3.2

采集设备 acquisition equipment

用于采集电梯运行状态、运行统计信息、故障信息和图像信息，具备识别、检测、转换、报警、通信及音视频播放功能的电子设备。

3.3

运行监测数据采集器 monitoring data acquisition

用于采集电梯运行状态、运行统计信息和故障信息的电子装置，它是采集设备的一部分。

3.4

图像数据采集器 image data acquisition

用于采集、处理电梯图像信息的电子装置，它是采集设备的一部分。

3.5

一体式数据采集器 integrated data acquisition

同时具备运行监测数据采集和图像数据采集功能的采集器。

3.6

长连接 long connection

在一个通信过程中连续发送多个数据包，如果没有数据包发送，需要采集设备发送心跳包以维持此连接，并作为监测链路的手段。

3.7

电梯钢丝绳运行折弯次数 number of lift steel wire rope running bending

电梯轿厢运行方向转换一次记为电梯钢丝绳运行折弯1次。

4 缩略语

下列缩略语适用于本部分：

RTP：实时传输协议 （Real-time Transport Protocol）

UDP：用户数据包协议 （User Datagram Protocol）

FTP：文件传输协议 （File Transfer Protocol）

NTP：时间同步协议 （Network Time Protocol）

5 通信协议

5.1 通信方式

5.1.1 协议

采用 UDP 作为传输层协议，要求如下：

——服务器端口：8800-8899 区间；

——所有保留字段都应当置 0x0；

——字节顺序采用大端模式(big-endian)。

示例：

普通数值数据 0x1234 = { byte[0] = 0x12; byte[1] = 0x34; }

IP 地址 192.168.103.9 = { byte[0]=192;byte[1] =168;byte[2]= 103;byte[3]=9;}

5.1.2 数据包

数据包由包头、数据实体和校验码组成，如图1所示。

包头	数据实体	校验码

图1 数据包结构示意图

注1：包头是每包数据报文必须符合的数据格式头，包括：Session ID，数据包流水号，数据类型，数据实体长度，

协议版本，保留字段。

注2：数据实体是具体数据内容。

注3：校验码采用CRC16校验方式对包头和数据实体进行校验所产生的结果码，参见附录A。

5.2 通信对象

本协议的通信对象是采集设备与服务器。

5.3 通信过程

由采集设备通过登录请求包登录到服务器，登录成功后通过采集设备向服务器发送心跳包来保持长连接。在长连接的条件下，采集设备与服务器可以完成数据传输和时钟同步等功能。见图2。

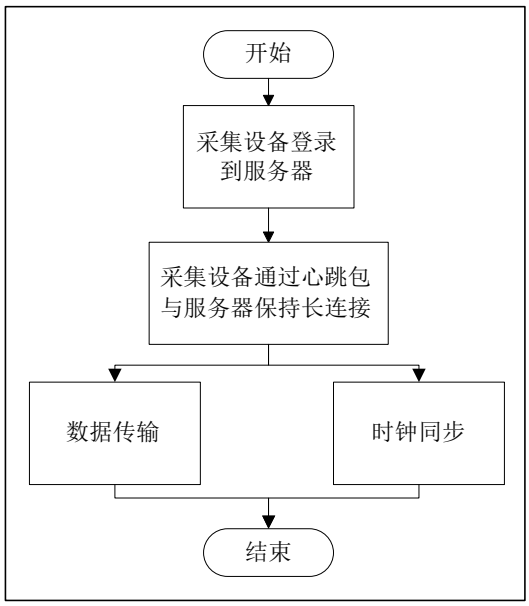


图2 通信过程图

5.4 通信时序

采集设备主动向服务器发起通信连接，用于双方信息的交互。当没有数据传输时，采集器应按周期发送心跳包以维持此连接，当心跳包发送10次后服务器返回一次心跳应答包。报文采用同步方式发送。见图3。

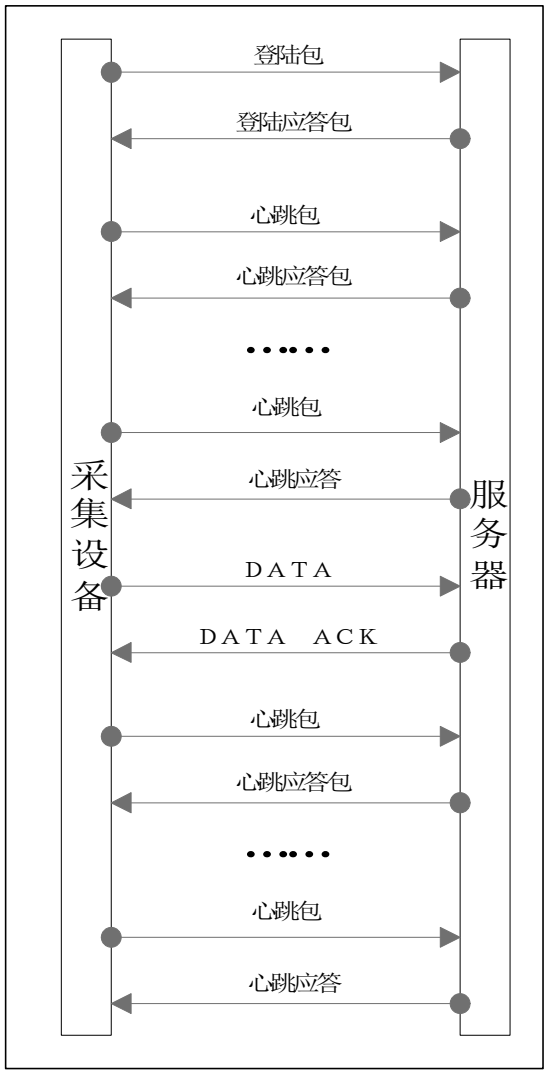


图3 通信时序图

5.5 通信连接的建立

5.5.1 运行监测数据采集器或一体式数据采集器登录服务器

5.5.1.1 运行监测数据采集器或一体式数据采集器在确认网络已连接的情况下，向服务器发送登录请求数据包。服务器收到登录请求数据包后，判断是否允许运行监测数据采集器或一体式数据采集器登录。处理方法如下：

——若运行监测数据采集器或一体式数据采集器身份数据合法，服务器向运行监测数据采集器或一体式数据采集器返回登录正确数据包，连接建立；

——若运行监测数据采集器或一体式数据采集器身份数据非法，服务器向运行监测数据采集器或一体式数据采集器返回登录失败数据包，运行监测数据采集器或一体式数据采集器应在 60 秒后重新尝试登录服务器。

5.5.1.2 若运行监测数据采集器或一体式数据采集器在设定的时间内没有收到服务器返回的登录请求应答包，运行监测数据采集器或一体式数据采集器需要重新尝试登录到服务器。

5.5.2 图像数据采集器登录服务器

5.5.2.1　图像数据采集器在确认网络已连接的情况下，向服务器发送图像数据采集器登录请求数据包。服务器收到图像数据采集器登录请求数据包后，判断是否允许图像数据采集器登录。处理方法如下：

——若图像数据采集器身份数据合法，服务器向图像数据采集器返回登录正确数据包，连接建立；

——若图像数据采集器身份数据非法，服务器向图像数据采集器返回登录失败数据包，图像数据采集器应在 60 秒后重新尝试登录服务器。

5.5.2.2　若图像数据采集器在设定的时间内没有收到服务器返回的图像数据采集器登录请求应答包，图像数据采集器需要重新尝试登录到服务器。

5.5.3　采集设备登录服务器流程见图 4。

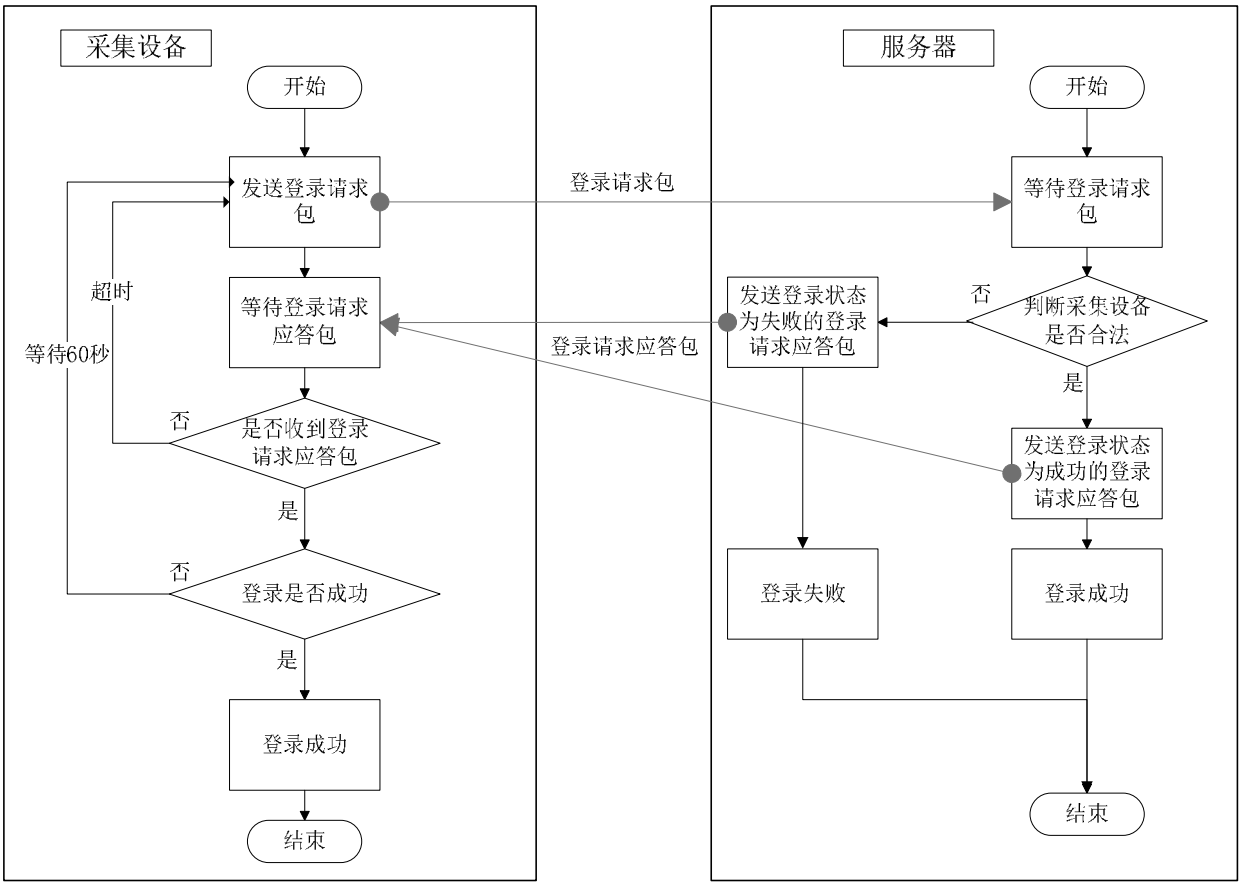


图4　采集设备登录服务器流程图

5.5.4　采集设备与服务器的连接保持

5.5.4.1　登录成功后，采集设备应主动给服务器发送心跳包来与服务器保持长连接，采集设备按设定的时间周期 T 向服务器发送心跳包。T 的取值范围：5 秒～15 秒。

5.5.4.2　服务器收到心跳包后应将采集设备信息记录下来，同时向采集设备回复心跳应答包。

5.5.4.3　当服务器在 10*T 秒内没有收到心跳包，认为通信链路中断，关闭会话。

5.5.4.4　当采集设备在连续发送了 30 个心跳包后，没有收到服务器的心跳应答包，认为通信链路中断，应重新登录服务器。

5.5.4.5　采集设备与服务器的连接保持流程见图 5。

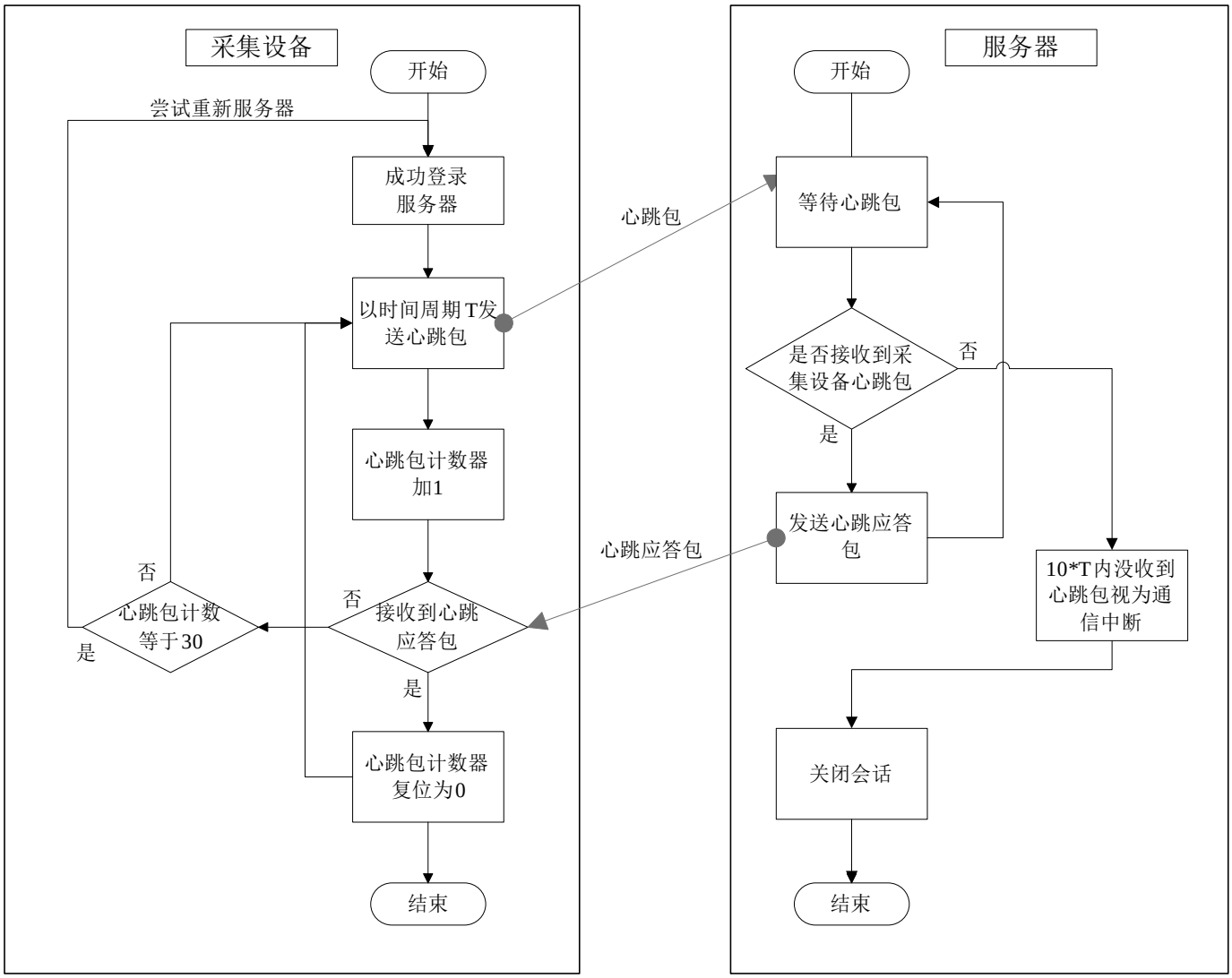


图5　采集设备与服务器的连接保持流程图

5.6　实时数据传输

5.6.1　运行状态数据传输

5.6.1.1　运行状态数据传输由服务器发起。

5.6.1.2　服务器按需向采集设备发送启动运行状态数据传输命令。采集设备收到该命令后，应按服务器要求采样周期采集数据，并按照服务器要求的传输周期和时限向指定服务器（指令内指定的 IP、端口号的服务器）发送运行状态数据包。

5.6.1.3　采集设备应按照下列条件停止发送运行状态数据包：

——当采集设备发送运行状态数据包超过命令设定的时限时，应停止发送；

——当采集设备判断通信链路中断时，应停止发送；

——当采集设备收到服务器发送的停止运行状态数据传输命令时，应停止发送。

5.6.1.4　运行状态数据传输流程见图 6。

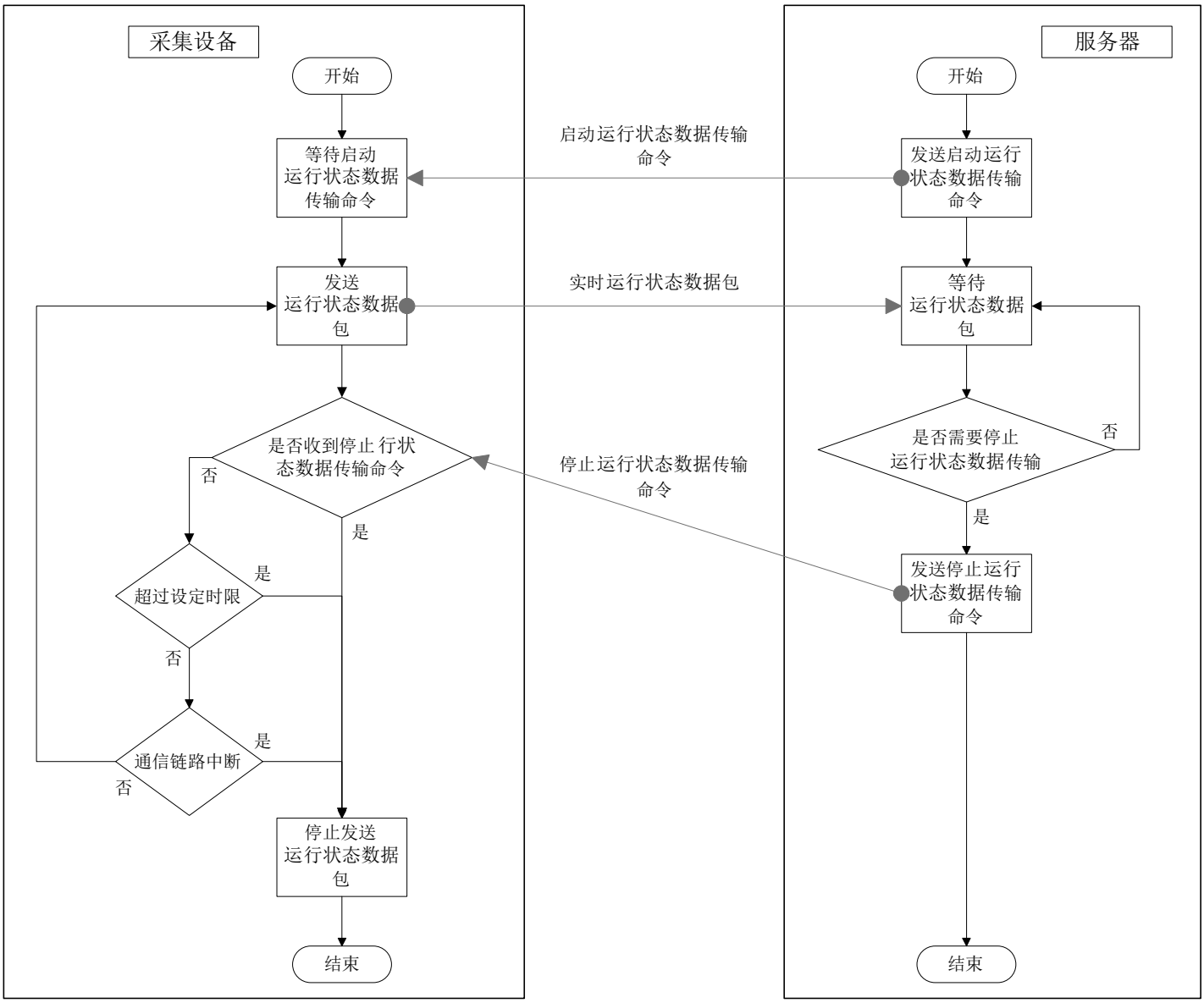


图6　运行状态数据传输流程图

5.6.2　故障数据传输

5.6.2.1　当采集设备判定故障情况发生变化时，包括故障发生、故障消除或故障中运行状态发生变化，应立刻向服务器发送故障数据包。

5.6.2.2　采集设备收到服务器的故障数据查询命令包时，应发送反映当前故障情况的故障数据包。

5.6.2.3　服务器收到故障数据包后应回复故障数据应答包。

5.6.2.4　采集设备 5 秒内没收到服务器故障数据应答包，应继续向服务器发送反映当前故障情况的故障数据包，直至收到服务器故障数据应答包。

5.6.2.5　若通信链路中断，应停止发送故障数据包，待通信链路恢复采集设备重新登录后，应上传反映当前故障情况的故障数据包。

5.6.2.6　故障数据传输流程见图 7。

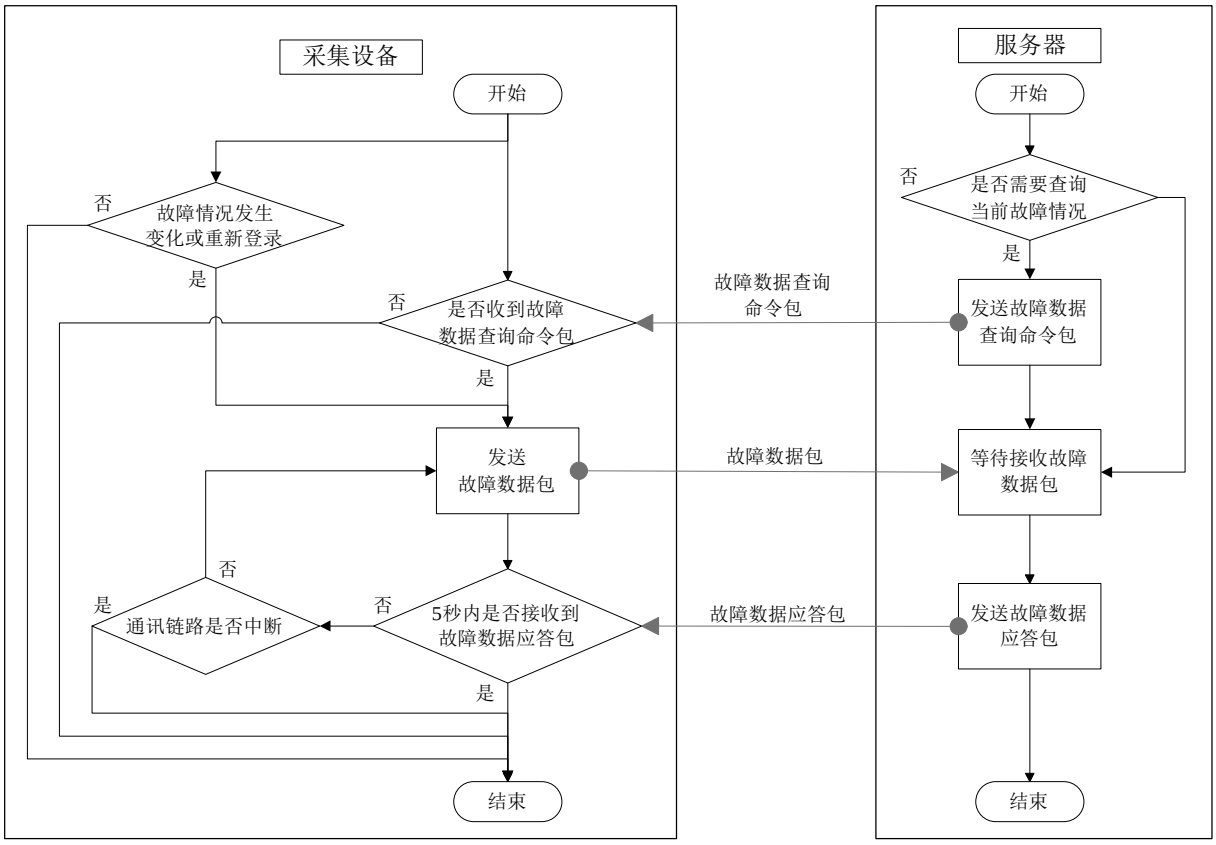


图7　故障数据传输流程图

5.6.3　实时图像数据传输

5.6.3.1　服务器按需向采集设备发送启动实时图像数据传输命令。采集设备收到该命令后，应按服务器要求，向指定服务器的指定端口发送图像编码参数数据包。服务器在收到该参数数据包后，向采集设备发送开始传输实时图像数据命令包。当服务器已知当前采集设备图像编码参数数据时，可直接发送开始传输实时图像数据命令，采集设备收到该命令后，应向指定 IP、端口号的服务器传输实时图像数据。

5.6.3.2　采集设备收到开始传输实时图像数据命令包后，应向指定 IP 和端口号的服务器发送实时图像数据包。如果没有相应的实时图像数据，应向指定 IP 和端口号的服务器发送关闭实时图像数据传输命令应答包。

5.6.3.3　如果采集设备在发送图像编码参数数据包 2 秒后，未收到开始传输实时图像数据命令，则重新发送图像编码参数数据包。发送图像编码参数数据包 5 次后，仍未收到开始传输实时图像数据命令，则应停止发送。

5.6.3.4　如果采集设备在重复发送图像编码参数数据包的过程中，收到了关闭实时图像数据传输命令包，应停止上传图像编码参数数据包。

5.6.3.5　服务器不需要采集设备发送实时图像数据包时，应向采集设备发送关闭实时图像数据传输命令，采集设备收到该命令后应停止实时图像数据传输，并发送关闭实时图像数据传输命令应答包到指定服务器。

5.6.3.6　当采集设备判断通信链路中断，应停止发送实时图像数据包。

5.6.3.7　实时图像数据传输流程见图 8。

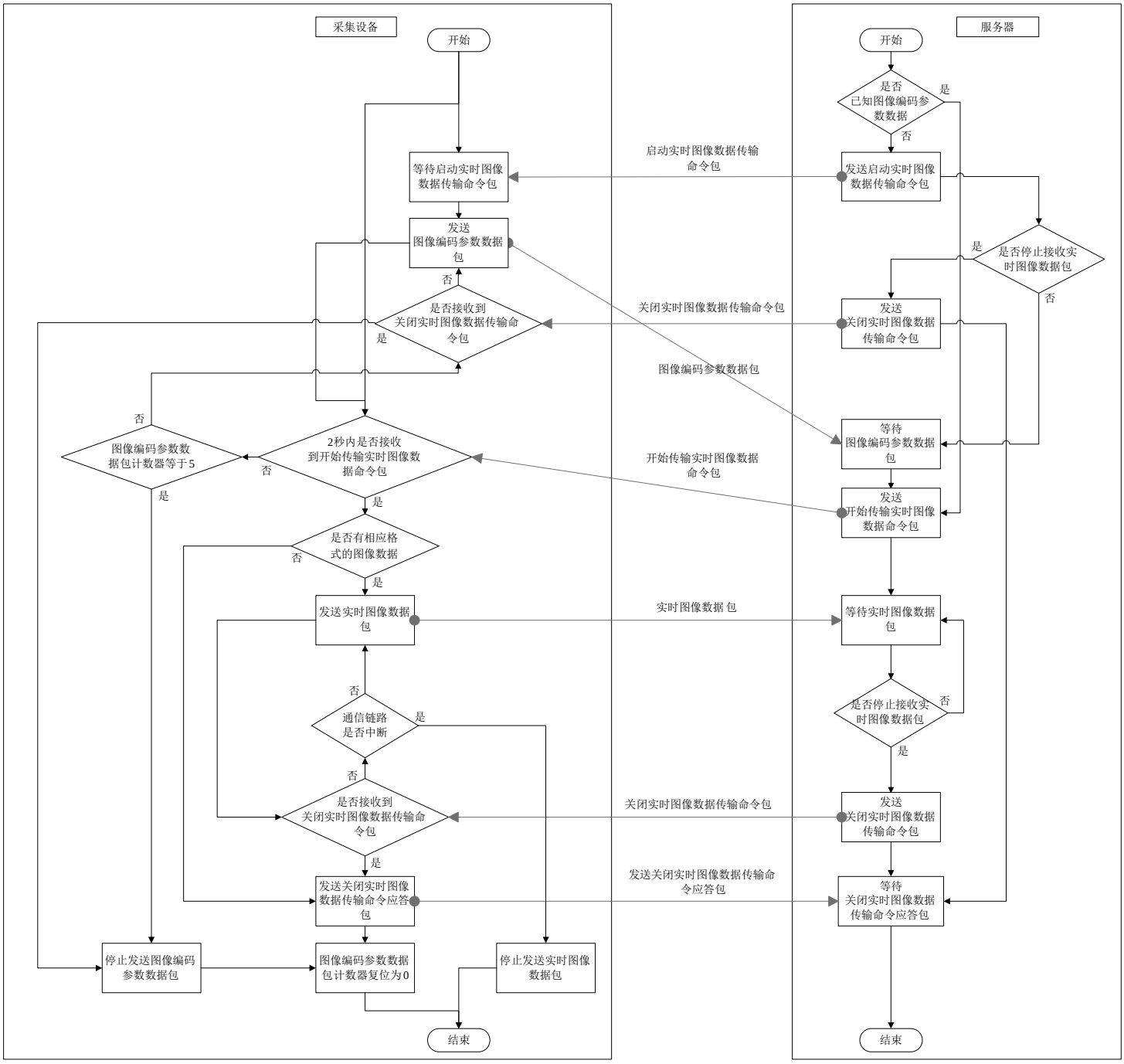


图8　实时图像数据传输流程图

5.7　运行统计数据传输

运行统计数据由采集设备定时上传。服务器不做应答。运行统计数据传输流程见图 9。

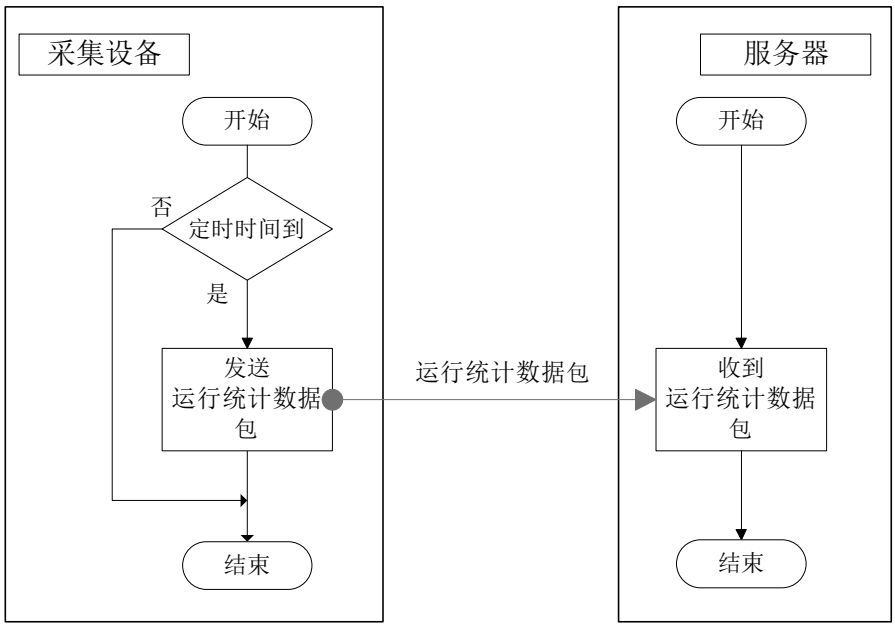


图9　运行统计数据传输流程图

5.8　文件数据传输

5.8.1　服务器按需向采集设备发送启动文件传输命令。

5.8.2　采集设备收到命令后，返回一个启动文件传输命令应答包，然后根据命令，通过 FTP 协议上传或下载文件。

5.8.3　当文件传输结束后，采集设备应向请求该文件的服务器发送文件传输结果反馈包，服务器收到文件传输结果反馈包后，应向采集设备发送文件传输结果反馈应答包。采集设备做如下处理：

——如果采集设备收到了服务器的文件传输结果反馈应答包，则停止发送文件传输结果反馈包；

——如果采集设备 5 秒内没有收到服务器的文件传输结果反馈应答包，则采集设备应重新发送文件传输结果反馈包到服务器。如果采集设备发送文件传输结果反馈包计数次数达到 10 次，则应停止发送文件传输结果反馈包。

5.8.4　异常文件传输流程处理如下：

——当采集设备正在向服务器传输文件时，收到服务器请求传输同一文件的启动文件传输命令，采集设备应发送文件传输结果反馈包，包中传输状态填写 FTP 代码；

——当文件 FTP 传输已成功完成时，收到服务器请求传输同一文件的启动文件传输命令，采集设备应重新传输该文件；

——在采集设备发送文件传输结果反馈包后，未收到服务器的文件传输结果反馈应答包前，收到服务器请求传输同一文件的启动文件传输命令，采集设备应直接发送文件传输结果反馈包，发送累计达到 10 次，应停止发送文件传输结果反馈数据包。

5.8.5　FTP 服务器端以及 FTP 客户端均应支持断点续传。

5.8.6　采集设备应按照服务器生成的“FTP 文件传输编号” 作为文件的唯一标识，判断是否为同一文件传输请求。

5.8.7　文件数据传输流程见图 10。

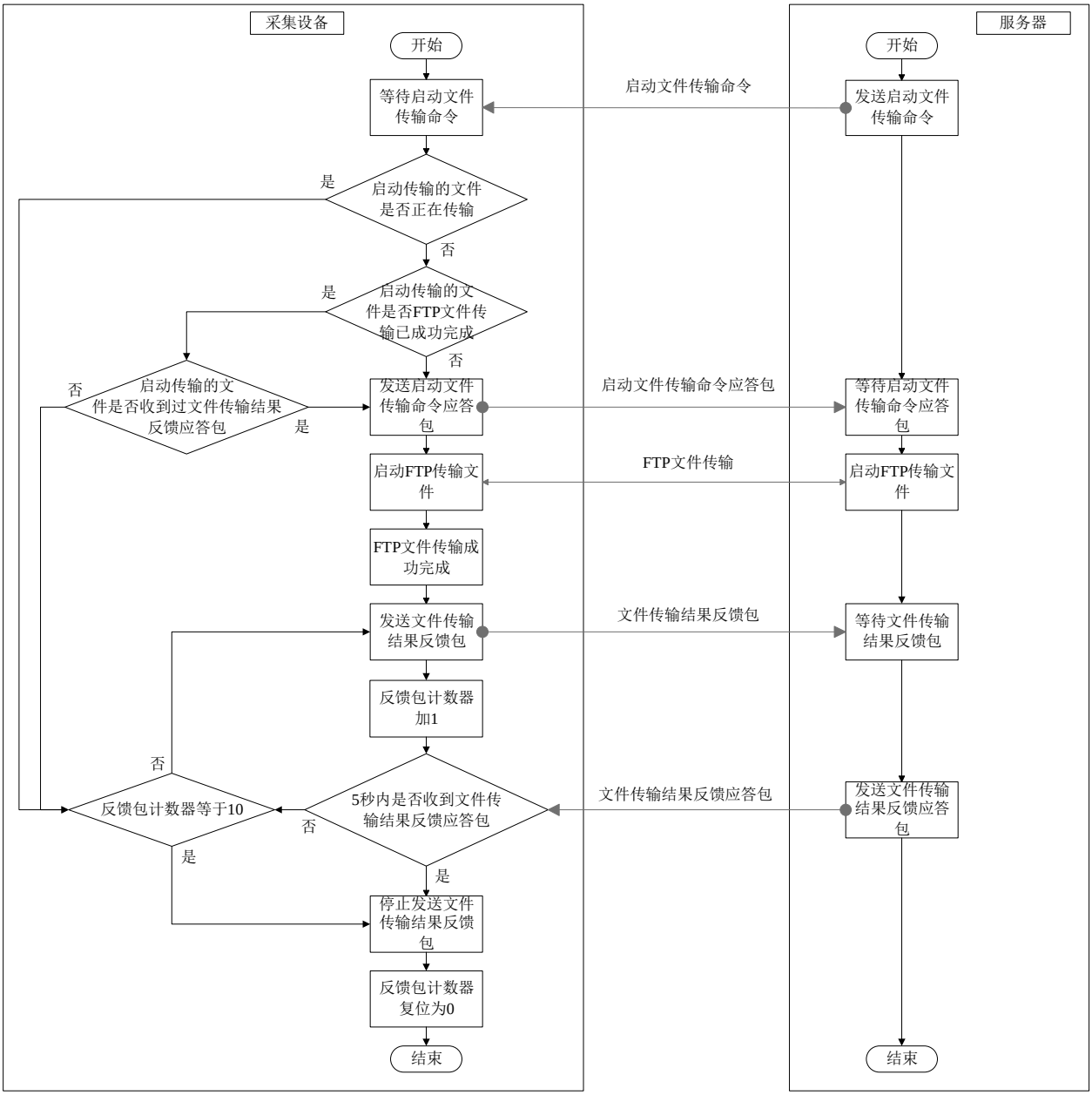


图10　文件数据传输流程图

5.9　时间同步

采集设备应与服务器实现时间同步。并按如下要求进行：

——应采用 NTP 协议，平台作为 NTP 服务器端；

——采集设备上电时，应做一次时间同步；

——当采集设备有需要时进行时间同步；

——NTP 协议参见附录 B。

6　数据格式

6.1 数据包结构

6.1.1 采集设备发送给服务器或应答服务器的数据包定义如表1。

表1 采集设备发送给服务器或应答服务器的数据包定义表

格式	通信字节	协议包（Packet）内容	长度
包头	BYTE0～17	Session ID	18BYTE
	BYTE18～19	数据包流水号（Number），达到最大值时归零	2BYTE
	BYTE20～21	数据类型（Type）	2BYTE
	BYTE22～23	数据实体长度（Length）	2BYTE
	BYTE24～25	协议版本（Version）	2BYTE
数据实体	BYTE26～n	数据实体（Data）	n-25 BYTE
校验码	BYTEn+1～n+2	CRC16 校验（校验范围为 BYTE0～BYTEn）	2BYTE

6.1.2 服务器发送给采集设备或应答采集器的数据包定义如表2。

表2 服务器发送给采集设备或应答采集器的数据包定义表

格式	通信字节	协议包（Packet）内容	长度
包头	BYTE0～1	数据包流水号（Number），达到最大值时归零	2BYTE
	BYTE2～3	数据类型（Type）	2BYTE
	BYTE4～5	数据实体长度（Length）	2BYTE
	BYTE6～7	协议版本（Version）	2BYTE
	BYTE8～11	保留字段	4BYTE
数据实体	BYTE12～n	数据实体（Data）	n-11 BYTE
校验码	BYTEn+1～n+2	CRC16 校验（校验范围为 BYTE0～BYTEn）	2BYTE

6.1.3 数据包结构说明如下：

——Session ID：采集设备登录服务器时，登录数据包的 Session ID 为 0。采集设备登录服务器成功后，服务器分配给采集设备的临时身份 ID；

——数据包流水号（Number）：数据包的顺序号，用来标识数据包发出的先后顺序；

——数据类型（Type)：标识是哪种类型的数据包，数据类型定义见表 3；

——数据实体长度（Length）：数据实体 Data 的长度；

——协议版本（Version）：协议的版本号（计算方法：4.2=(byte[0]=4;byte[1]=2;)）；

——数据实体（Data）：包含具体的数据，由不同类型的数据包具体决定；

——CRC 校验：16 位循环冗余校验算法，校验范围为 BYTE0～BYTEn。

表3 数据类型定义

序号	数据类型(TYPE)	数据类型编码	发起端
1	登录请求	0x8001	采集设备
2	登录请求应答	0x9001	服务器端
3	心跳	0x8002	采集设备
4	心跳应答	0x9002	服务器端
5	启动/停止运行状态数据传输命令	0x9003	服务器端

6	运行状态数据	0x8003	采集设备
7	故障数据	0x8004	采集设备
8	故障数据应答	0x9004	服务器端
9	故障数据查询命令	0x9005	服务器端
10	启动文件传输命令	0x9006	服务器端
11	启动文件传输命令应答	0x8006	采集设备
12	文件传输结果反馈	0x8007	采集设备
13	文件传输结果反馈应答	0x9007	服务器端
14	运行统计数据	0x8008	采集设备
15	运行统计数据应答	0x9008	服务器端
16	图像数据采集器登录请求	0x8009	图像数据采集设器
17	图像数据采集器登录请求应答	0x9009	服务器端
18	启动/关闭实时图像数据传输命令	0x9010	服务器端
19	图像编码参数数据	0x8010	采集设备
20	开始传输实时图像数据命令	0x9011	服务器端
21	实时图像数据	0x8011	采集设备
22	关闭实时图像数据传输命令应答	0x8012	采集设备

6.2 数据实体定义

6.2.1 登录请求（0x8001）数据实体见表 4。

表4 采集设备登录请求数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0～1	采集设备分类编码 BCD 码 0～9999（大端模式） 详见表 5	由采集设备生产厂家出厂时设定，在安装启用后与电梯信息一起录入服务器，作为注册信息，服务器收到采集设备登录请求后，比对请求包中的信息与服务器中的注册信息是否一致，判断是否合法	整型/2BYTE
BYTE2～7	采集设备 MAC 地址（大端模式）		整型/6BYTE
BYTE8～35	采集设备编码，ASCII 码，详见表 6		字符串/28BYTE
BYTE36～38	采集设备软件版本		字符串/3BYTE
BYTE39～40	预留		整型/2BYTE
BYTE41～44	生产日期（BCD 码） YYYY-MM-DD		整型/4BYTE
BYTE45～52	保留		整型/8BYTE
BYTE53～56	数据生成的时间戳（从 1970 年 1 月 1 日 0 时 0 秒到当前时间的总秒数，UTC 时间）	服务器根据时间相应采集设备的登录请求	整型/4BYTE

表5　采集设备分类编码

设备名称	编码
运行监测数据采集器	0001
图像数据采集器	0002
一体式数据采集器	0003

表6　采集设备编码

标识字符区域	区域说明
1～9	组织机构代码
10～28	厂家自定义编码
注： a)　该字段为字符串。字符串约束范围：0～9、A～Z。 b)　编码数据格式	

6.2.2　登录请求应答（0x9001）数据实体见表 7。

表7　登录请求应答数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	登录状态	采集设备判定是否继续发送登录请求	整型/1BYTE
BYTE1～18	临时身份 ID	服务器确认采集设备是否合法	整型/18 BYTE
BYTE19～27	保留	保留	整型/9BYTE
BYTE28～31	时间（秒）	采集设备同步服务器时间	整型/4BYTE

6.2.2.1　登录成功，则应答临时身份 ID、时间；

6.2.2.2　登录失败，则只返回登录状态。

6.2.2.3　登录状态代码如下：

——0x01：登录成功；

——0x02：登录失败；

——0x08：服务器中无此采集设备的信息。

6.2.3　心跳（0x8002）数据实体见表 8。

表8　心跳数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	计数器值	采集设备根据计数器值判定是否服务器掉线	整型/1BYTE

6.2.4　心跳应答（0x9002）数据实体定义：空。

6.2.5　启动/停止运行状态数据传输命令（0x9003）数据实体见表 9。

表9　启动或停止运行状态数据传输命令数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	运行状态数据传输开关	服务器查看运行状态数据开或关，0：表示停止运行状态数据传输；1：表示开始运行状态数据传输	整型/1BYTE

BYTE1～2	运行状态数据采样周期	用来控制采集设备采集运行状态数据周期。默认为 0，表示以采集设备默认采样周期采样。非 0 表示采样周期，单位为毫秒。(默认 500 毫秒采集一次，最小 100 毫秒)	整型/2BYTE
BYTE3～4	运行状态数据传输周期	0：表示采集设备默认的传输方式，即智能传输；(默认 2 秒传输一次，最小 100 毫秒，该值为采样周期的整数倍) 非 0：传输间隔时间，单位为毫秒。	整型/2BYTE
BYTE5	时限	0：无时限；其它：分钟	整型/1BYTE
BYTE6～9	IP 地址	运行状态数据采集服务器 IP 地址	整型/4BYTE
BYTE10～11	端口号	运行状态数据采集服务器端口号	整型/2BYTE

6.2.6 运行状态数据（0x8003）数据实体见表 10。

表10 运行状态数据实体定义

序号	电梯信号	自动扶梯和自动人行道信号	备注
BYTE0～1	电梯当前层站信息	保留	电梯控制系统输出或采集设备判定
BYTE2.bit0～1	电梯运行方向信号，bit1，bit0 = 00表示停止；=01表示下行；=10表示上行；=11无意义	运行方向信号，bit1，bit0 = 00表示停止；=01表示下行；=10表示上行；=11无意义	电梯控制系统输出或采集设备判定
BYTE2.bit2～3	电梯开关门状态信号，bit3,bit2=00表示开门状态；11表示关门状态。	保留	电梯控制系统输出或采集设备判定
BYTE2.bit4	电梯检修状态信号，=0表示正常；=1表示检修	检修状态信号，=0表示正常；=1表示检修	电梯控制系统输出
BYTE2.bit5	电梯运行状态信号，=0表示正常；=1表示故障	运行状态信号，=0表示正常；=1表示故障	电梯控制系统输出
BYTE2.bit6	电梯供电中断信号，=0表示正常；=1表示故障	供电中断信号	采集设备判定
BYTE2.bit7	轿厢内乘客感应信号,=0表示正常，=1表示有人	保留	采集设备判定
BYTE3.bit0	消防服务信号，=0表示非消防模式；=1表示消防模式	保留	电梯控制系统输出
BYTE3.bit1	有司机服务信号，=0表示无司机服务模式；=1表示有司机服务模式	保留	电梯控制系统输出
BYTE3.bit2～7	保留	保留	
BYTE4～7	数据生成的时间戳（从1970年1月1日0时0秒到当前时间的总秒数，UTC时间）（定时器时间的秒总数）	秒时间戳	

序号	电梯信号	自动扶梯和自动人行道信号	备注
BYTE8～11	微秒时间戳（定时器时间去除秒后的剩余值） 例：定时器时间总数为1576800000 100300微秒 秒时间戳为：1576800000 微秒时间戳为：100300	微秒时间戳	

6.2.7 故障数据（0x8004）数据实体见表 11。

故障数据实体中的反映故障的数据用代码表示，反映故障发生时电梯运行状态的数据以位表示。

表11 故障数据实体定义

序号	电梯信号	自动扶梯和自动人行道信号	备注
BYTE0～1	电梯所在层站信号	保留	电梯控制系统输出或采集设备判定
BYTE2.bit0～1	电梯运行方向信号，bit1，bit0 = 00表示停止；=01表示下行；=10表示上行；=11无意义	运行方向信号，bit1，bit0 = 00表示停止；=01表示下行；=10表示上行；=11无意义	电梯控制系统输出或采集设备判定
BYTE2.bit2～3	电梯开关门状态信号，bit3,bit2=00表示开门状态；11表示关门状态。	保留	电梯控制系统输出或采集设备判定
BYTE2.bit4	电梯检修状态信号，=0表示正常；=1表示检修	检修状态信号，=0表示正常；=1表示检修	电梯控制系统输出
BYTE2.bit5	电梯运行状态信号，=0表示正常；=1表示故障	运行状态信号，=0表示正常；=1表示故障	电梯控制系统输出
BYTE2.bit6	电梯供电中断信号，=0表示正常；=1表示故障	供电中断信号，=0表示正常；=1表示故障	采集设备判定
BYTE2.bit7	轿厢内乘客感应信号，=0表示无人；=1表示有人	保留	采集设备判定
BYTE3.bit0	消防服务信号，=0表示非消防模式；=1表示消防模式	保留	电梯控制系统输出
BYTE3.bit1	有司机服务信号，=0表示无司机服务模式；=1表示有司机服务模式	保留	电梯控制系统输出
BYTE3.bit2	采集设备检修状态，=0表示正常；=1表示检修		
BYTE3.bit3	采集设备与电梯控制系统通信异常，=0表示正常；=1表示异常		
BYTE3.bit4	采集设备故障，=0表示正常；=1表示有故障。		
BYTE3.bit5～7	保留	保留	
BYTE4～7	电梯运行时间 (长整型 4 个字节)，单位：分钟		
BYTE8～11	电梯运行次数(长整型 4 个字节)		
BYTE12～12+n*2	故障码，长度＝n*2(n 为故障代码数量)		
BYTE(13＋n*2)～(13＋n*2+4)	数据生成的时间戳（从 1970 年 1 月 1 日 0 时 0 秒到当前时间的总秒数，UTC 时间），长整型 4 个字节，单位：秒		

注1：故障代码详见附录C。

6.2.8 故障数据应答包（0x9004）数据实体定义：空。

6.2.9 故障数据查询命令（0x9005）数据实体（Data）定义：空。

6.2.10 启动文件传输命令（0x9006）数据实体定义见表12。

表12 启动文件传输命令数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	传输方式	采集设备判定上传或下载	整型/1BYTE
BYTE1	文件类型	采集设备判定上传文件或下载存储文件类型	整型/1BYTE
BYTE2～9	文件信息	若为上传图像文件：BYTE2～BYTE5 为起始时间，BYTE6～9 为结束时间，单位为秒。	整型/8BYTE
BYTE10～13	FTP 服务器 IP 地址	采集设备登录 FTP 服务器地址（如：C0 A8 08 64 即：192.168.8.100）	整型/4BYTE
BYTE14～15	FTP 服务器端口		整型/2BYTE
BYTE16～35	FTP 服务器用户名	登录 FTP 用户名（不足 20 字节则左对齐，右补空格）	字符串/20 BYTE
BYTE36～45	FTP 服务器密码	登录 FTP 密码(不足 10 字节则左对齐，右补空格)	字符串/10BYTE
BYTE46～95	文件路径	FTP 文件路径(不足 50 字节则左对齐，右补空格)	字符串/50 BYTE
BYTE96～125	文件名	FTP 文件名(不足 30 字节则左对齐，右补空格)	字符串/30 BYTE
BYTE126～157	FTP 文件传输编号	作为当前 FTP 文件传输的唯一标识	字符串/32BYTE

6.2.10.1 传输方式代码如下：

——0x01：上传文件；
——0x02：下载文件。

6.2.10.2 **文件类型**

文件类型分上传和下载两类，具体代码如下：
——上传：

- 0x01：历史图像文件；
- 0x02：故障图像片段文件；
- 0x03：运行状态历史文件；
- 0x04：当前多媒体列表文件。

——下载：

- 0x10：运行监测数据采集器软件更新文件；

- 0x11：图像数据采集器软件更新文件；
- 0x12：一体式数据采集器软件更新文件；
- 0x13：多媒体文件；
- 0x14：多媒体播放列表文件。

6.2.11 启动文件传输命令应答（0x8006）数据实体定义见表 13。

表13 启动文件传输命令应答数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0～31	FTP 文件传输编号	作为当前 FTP 文件传输的唯一标识	字符串/32BYTE

6.2.12 文件传输结果反馈（0x8007）数据实体定义见表 14。

表14 文件传输结果反馈数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0～31	FTP 文件传输编号	作为当前 FTP 文件传输的唯一标识	字符串/32BYTE
BYTE32～35	传输状态	文件传输状态代码	整型/4BYTE
注 1：0x00000000：文件传输成功完成 注 2：0x00000001：解密失败 注 3：0x00000xxx：FTP 代码。如 504 错误，则为 0x000001F8 注 4：文件传输成功完成和解密失败，使用自定义编码。其它传输状态使用 FTP 错误代码。			

6.2.13 文件传输结果反馈应答（0x9007）数据实体定义见表 15。

表15 文件传输结果反馈应答数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0～31	FTP 文件传输编号	作为当前 FTP 文件传输的唯一标识	字符串/32BYTE

6.2.14 运行统计数据包（0x8008）数据实体定义见表 16。

表16 运行统计数据实体定义

字节	数据实体
BYTE0～3	电梯运行时间，大端模式。单位：分钟
BYTE4～7	电梯运行次数，大端模式
BYTE8～11	电梯钢丝绳折弯次数，大端模式

6.2.15 运行统计数据包应答包（0x9008）数据实体（Data）定义：空。

6.2.16 图像数据采集器登录请求（0x8009）数据实体定义见表 17。

表17 图像数据采集器登录请求数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0～1	采集设备分类编码 BCD 码 0～9999（大端模式）	由采集设备生产厂家出厂时设定，在安装启用后与电梯信息一起录入服	整型/2BYTE

BYTE2～7	采集设备 MAC 地址（大端模式）	务器，作为注册信息，服务器收到采集设备登录请求后，比对请求包中的信息与服务器中的注册信息是否一致，判断是否合法	整型/6BYTE
BYTE8～35	图像数据采集器编码		字符串/28BYTE
BYTE36～39	生产日期（BCD 码） YYYY-MM-DD		整型/4BYTE
BYTE40～47	保留		整型/8BYTE
BYTE48～51	数据生成的时间戳（从 1970 年 1 月 1 日 0 时 0 秒到当前时间的总秒数，UTC 时间）	服务器根据时间相应采集设备的登录请求	整型/4BYTE
BYTE52	图像数据采集器视频采集通道数	标识图像数据采集器已使用的通道数	整型/1BYTE
BYTE53～ (52+n*28)	图像数据采集器通道对应该电梯运行监测数据采集器编码	图像数据采集器通道所对应的电梯运行监测数据采集器编码	整型/n*28BYTE

6.2.17 图像数据采集器登录请求应答（0x9009）数据实体定义见表 18。

表18 图像数据采集器登录请求应答数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	登录状态	图像数据采集器判定是否继续发送登录请求	整型/1BYTE
BYTE1～18	临时身份 ID	服务器确认图像数据采集器是否合法	整型/18 BYTE
BYTE19～27	保留	保留	整型/9BYTE
BYTE28～31	时间（秒）	图像数据采集器同步服务器时间	整型/4BYTE

6.2.17.1 登录成功，则应答临时身份 ID、时间；

6.2.17.2 登录失败，则只返回登录状态。

6.2.17.3 登录状态代码如下：

——0x01：登录成功；

——0x02：登录失败；

——0x05：在服务器中没有通道对应的运行监测数据采集器的信息；

——0x08：服务器中无此图像数据采集器的信息。

6.2.18 启动/关闭实时图像数据传输命令（0x9010）数据实体定义见表 19。

表19 启动/关闭实时图像数据传输命令数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	图像传输开关	服务器查看开或关 0：表示关闭图像传输 1：表示开启图像传输	整型/1BYTE

BYTE1～4	IP 地址		整型/4BYTE
BYTE5～6	端口号（Port）		整型/2BYTE
BYTE7	图像类型	采集设备上传图像类型	整型/1BYTE
BYTE8～BYTE35	运行监测数据采集器或一体式数据采集器编码	图像数据采集器通道所对应的电梯运行监测数据采集器编码或一体式数据采集器编码	字符串/28BYTE

6.2.18.1 图像传输开关代码如下：

——0x00：关闭实时图像数据传输；

——0x01：启动实时图像数据传输。

6.2.18.2 图像类型代码如下：

——0x01：QCIF 分辨率的图像（176*144）；

——0x02：CIF 分辨率的图像（352*288）；

——0x03：DCIF 分辨率的图像（528*384）；

——0x04：4CIF 分辨率的图像（704*576）。

6.2.19 图像编码参数数据包（0x8010）数据实体定义见表 20。

表20 图像编码参数数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	图像类型	采集设备上传图像类型	整型/1BYTE
BYTE1～2	帧率(每秒帧个数)		整型/2BYTE
注：图像类型： ——0x01：QCIF 分辨率的图像（176*144） ——0x02：CIF 分辨率的图像（352*288） ——0x03：DCIF 分辨率的图像（528*384） ——0x04：4CIF 分辨率的图像（704*576）			

6.2.20 开始传输实时图像数据命令（0x9011）数据实体定义见表 21。

表21 开始传输实时图像数据命令数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0～3	IP 地址	采集设备向该地址发送图像数据	整型/4BYTE
BYTE4	端口号		整型/2BYTE
BYTE5			
BYTE6	图像类型	采集设备上传图像类型	整型/1BYTE
BYTE7～BYTE34	运行监测数据采集器或一体式数据采集器编码	图像数据采集器通道所对应的电梯运行监测数据采集器编码或一体式数据采集器编码	字符串/28BYTE
注 1：图像类型： ——0x01：QCIF 分辨率的图像（176*144） ——0x02：CIF 分辨率的图像（352*288）			

——0x03：DCIF 分辨率的图像（528*384）
——0x04：4CIF 分辨率的图像（704*576）

6.2.21 实时图像数据包（0x8011）数据实体定义见表 22。

表22 实时图像数据包实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0～3	数据生成的时间戳（从 1970 年 1 月 1 日 0 时 0 秒到当前时间的总秒数，UTC 时间）	判断是否是同一帧数据和图像显示的先后，并且控制帧队列数据的丢弃；还可用于控件时间显示	整型/4BYTE
BYTE4～7	微秒时间戳		整型/4BYTE
BYTE8～11	预留	预留	整型/4BYTE
BYTE12～BYTEn	图像流（RTP 打包 H.264 图像数据包）	客户端恢复为图像	字符串/n-11BYTE

6.2.22 关闭实时图像数据传输命令应答（0x8012）数据实体定义见表 23。

表23 关闭实时图像数据传输命令应答数据实体定义

字节	数据实体	数据用途	数据类型/长度
BYTE0	图像传输状态	标识图像流当前传输状态	整型/1BYTE
BYTE1	指令执行状态	“启动/关闭实时图像数据传输”执行状态	整型/1BYTE
BYTE2～BYTE(1+n*28)	运行监测数据采集器或一体式数据采集器编码	图像数据采集器通道所对应的电梯运行监测数据采集器编码或一体式数据采集器编码	字符串/(n*28)BYTE
注 1：图像传输状态 ——0x00 图像传输停止 ——0x01 图像传输中 注 2：指令执行状态 ——0x01 图像类型不支持 ——0x02 无图像 ——0x03 其他			

6.2.23 **故障图像片段文件要求**

6.2.23.1 上传到服务器的图像文件格式应符合 H.264 文件标准规范，文件扩展名为 MP4。

6.2.23.2 故障图像片段时间=故障前 15 分钟+故障中（从故障开始到故障恢复的时间段）+故障结束后 15 分钟。

6.2.24 运行状态历史文件格式及存储要求如下：

——每天保存一个文件，文件名命名格式为：YYYYMMDD.TXT。YYYYMMDD 表示文件生成的日期，文

件名称统一使用大写。YYYYMMDD 均用阿拉伯数字表示，YYYY 表示年，MM 表示月，DD 表示天，如：20120606.TXT 表示 2012 年 6 月 6 日保存的文件；

——文件格式为文本文件，每行为一条记录，记录结束以回车（0x0D）表示。每条记录格式见表 24；

——最少保存 30 天的文件。

表24　运行状态历史文件记录定义

记录发生时间	运行状态	故障情况	记录结束符
hh:mm:ss	详见表 12 中 Byte0～3 的内容	详见表 12 中 BYTE12～12+n*2 的内容	回车(0x0D)

示例：采集设备中有一个文件 20110301.TXT，用文本格式打开后其内容为：

10:16:46 02 10 11 0 0 0 1 0000

10:16:47 03 10 11 0 0 0 1 0000

……

10:19:21 19 10 11 0 1 0 1 1301

附 录 A
（资料性附录）
CRC-16 校验代码

```
/*
//CRC 协议中给出的 CRC 校验程序
// 高位字节的 CRC 值
static unsigned char auchCRCHi[] =
{
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
};
// 低位字节的 CRC 值
static char auchCRCLo[] =
{
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04,
0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08,
0xC8,
0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C,
0xDC,
0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10,
0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C,
0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0,
0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8,
```

```
0x68,
    0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC,
0x7C,
    0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
    0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54,
    0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
    0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40
    };
    //函数以 unsigned short 类型返回 CRC
    //unsigned char *puchMsg ; // 用于计算 CRC 的报文
    //unsigned short usDataLen ; // 报文中的字节数
    unsigned short CRC16 (unsigned char * puchMsg,unsigned short usDataLen )
    {
        unsigned char uchCRCHi = 0xFF ;  // CRC 的高字节初始化
        unsigned char uchCRCLo = 0xFF ;  // CRC 的低字节初始化
        unsigned uIndex ;  // CRC 查询表索引
        while (usDataLen--)  // 完成整个报文缓冲区
        {
            uIndex = uchCRCLo ^ *puchMsg++ ;  // 计算 CRC
            uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex];
            uchCRCHi = auchCRCLo[uIndex] ;
        }
        return (uchCRCHi << 8 | uchCRCLo) ;
    }
    */
    (2) 移位运算
    UINT16   CRC16(UCHAR8 *Buf, UINT16 Lenth)
    {
        UINT16 CrcValueTmp;
        UINT16 j,i;
        CrcValueTmp = 0xffff;
        for(i=0; i<Lenth; i++)
        {
            CrcValueTmp ^= Buf[i];
            for(j=0; j<8; j++)
            {
                if(CrcValueTmp&0x0001)
                {
                    CrcValueTmp = ((CrcValueTmp>>1)^0xa001);
                }
                else
                {
```

```
                    CrcValueTmp >>= 1;
                }
            }
        }
        CrcValueTmp = (CrcValueTmp>>8) + (CrcValueTmp<<8); //低位在前，高位在后
        return CrcValueTmp;
}
```

附 录 B
（资料性附录）
NTP 协议客户端代码参考

NTP协议全称网络时间协议（Network Time Protocol）。它的目的是在国际互联网上传递统一、标准的时间。具体的实现方案是在网络上指定若干时钟源网站，为用户提供授时服务，并且这些网站间应该能够相互比对，提高准确度。

协议：UDP

端口：123

NTP 协议数据包格式

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
LI		VN			Mode				Stratum							Poll								Precision							
Root Dispersion																															
Reference Identifier																															
Reference Timestamp(64)																															
Originate Timestamp(64)																															
Receive Timestamp(64)																															
Transimit Timestamp(64)																															
Key Identifier(optional)																															
Message Digest(optional)																															

NTP 协议数据包各字段的含义

LI：跳跃指示器，警告在当月最后一天的最终时刻插入的迫近闰秒（闰秒）。

VN：版本号。

Mode：工作模式。该字段包括以下值：0－预留；1－对称行为；3－客户机；4－服务器；5－广播；6－NTP 控制信息。NTP 协议具有 3 种工作模式，分别为主/被动对称模式、客户/服务器模式、广播模式。在主/被动对称模式中，有一对一的连接，双方均可同步对方或被对方同步，先发出申请建立连接的一方工作在主动模式下，另一方工作在被动模式下；客户/服务器模式与主/被动模式基本相同，惟一区别在于客户方可被服务器同步，但服务器不能被客户同步；在广播模式中，有一对多的连接，服务器不论

客户工作在何种模式下，都会主动发出时间信息，客户根据此信息调整自己的时间。

Stratum：对本地时钟级别的整体识别。

Poll：有符号整数表示连续信息间的最大间隔。

Precision：有符号整数表示本地时钟精确度。

Root Delay：表示到达主参考源的一次往复的总延迟，它是有 15～16 位小数部分的符号定点小数。

Root Dispersion：表示一次到达主参考源的标准误差，它是有 15～16 位小数部分的无符号定点小数。

Reference Identifier：识别特殊参考源。

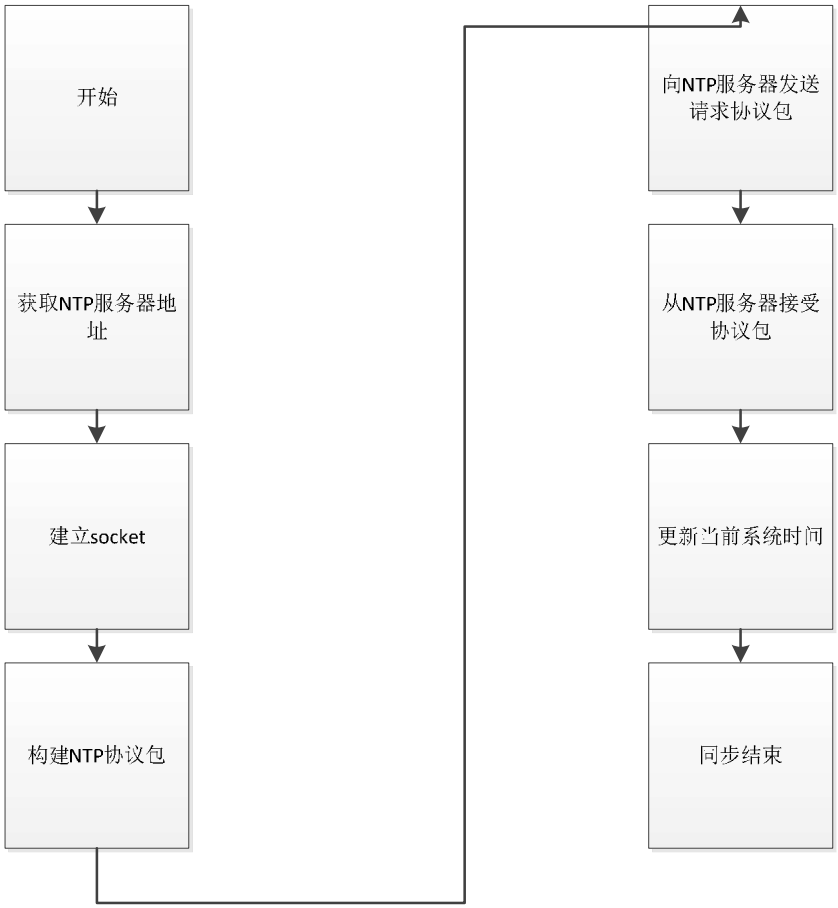
Originate Timestamp：这是向服务器请求分离客户机的时间，采用 64 位时标格式。

Receive Timestamp：这是向服务器请求到达客户机的时间，采用 64 位时标格式。

Transmit Timestamp：这是向客户机答复分离服务器的时间，采用 64 位时标格式。

Authenticator（Optional）：当实现了 NTP 认证模式时，主要标识符和信息数字域就包括已定义的信息认证代码（MAC）信息。

NTP 客户端流程



参考代码

```
/* ntp.c */
#include <sys/socket.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/un.h>
#include <sys/time.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <netinet/in.h>
#include <string.h>
#include <netdb.h>

#define NTP_PORT            123          /*NTP 专用端口号字符串*/
#define TIME_PORT           37           /* TIME/UDP 端口号 */
#define NTP_SERVER_IP       "210.72.145.44" /*国家授时中心 IP*/
#define NTP_PORT_STR        "123"        /*NTP 专用端口号字符串*/
#define NTPV1               "NTP/V1"     /*协议及其版本号*/
#define NTPV2               "NTP/V2"
#define NTPV3               "NTP/V3"
#define NTPV4               "NTP/V4"
#define TIME                "TIME/UDP"

#define NTP_PCK_LEN 48
#define LI 0
#define VN 3
#define MODE 3
#define STRATUM 0
#define POLL 4
#define PREC -6

#define JAN_1970 0x83aa7e80 /* 1900 年～1970 年之间的时间秒数 */
#define NTPFRAC(x)     (4294 * (x) + ((1981 * (x)) >> 11))
#define USEC(x)        (((x) >> 12) - 759 * ((((x) >> 10) + 32768) >> 16))

typedef struct _ntp_time
{
   unsigned int coarse;
   unsigned int fine;
```

```
} ntp_time;

struct ntp_packet
{
    unsigned char leap_ver_mode;
    unsigned char startum;
    char poll;
    char precision;
    int root_delay;
    int root_dispersion;
    int reference_identifier;
    ntp_time reference_timestamp;
    ntp_time originage_timestamp;
    ntp_time receive_timestamp;
    ntp_time transmit_timestamp;
};

char protocol[32];
/*构建 NTP 协议包*/
int construct_packet(char *packet)
{
    char version = 1;
    long tmp_wrd;
   int port;
    time_t timer;
    strcpy(protocol, NTPV3);
    /*判断协议版本*/
    if(!strcmp(protocol, NTPV1)||!strcmp(protocol, NTPV2)
        ||!strcmp(protocol, NTPV3)||!strcmp(protocol, NTPV4))
    {
        memset(packet, 0, NTP_PCK_LEN);
        port = NTP_PORT;
        /*设置 16 字节的包头*/
        version = protocol[6] - 0x30;
        tmp_wrd = htonl((LI << 30)|(version << 27)
            |(MODE << 24)|(STRATUM << 16)|(POLL << 8)|(PREC & 0xff));
       memcpy(packet, &tmp_wrd, sizeof(tmp_wrd));

        /*设置 Root Delay、Root Dispersion 和 Reference Indentifier */
        tmp_wrd = htonl(1<<16);
        memcpy(&packet[4], &tmp_wrd, sizeof(tmp_wrd));
        memcpy(&packet[8], &tmp_wrd, sizeof(tmp_wrd));
        /*设置 Timestamp 部分*/
```

```
            time(&timer);
            /*设置 Transmit Timestamp coarse*/
            tmp_wrd = htonl(JAN_1970 + (long)timer);
            memcpy(&packet[40], &tmp_wrd, sizeof(tmp_wrd));
            /*设置 Transmit Timestamp fine*/
            tmp_wrd = htonl((long)NTPFRAC(timer));
            memcpy(&packet[44], &tmp_wrd, sizeof(tmp_wrd));
            return NTP_PCK_LEN;
        }
        else if (!strcmp(protocol, TIME))/* "TIME/UDP" */
        {
            port = TIME_PORT;
            memset(packet, 0, 4);
            return 4;
        }
        return 0;
    }
    /*获取 NTP 时间*/
    int get_ntp_time(int sk, struct addrinfo *addr, struct ntp_packet *ret_time)
    {
        fd_set pending_data;
        struct timeval block_time;
        char data[NTP_PCK_LEN * 8];
        int packet_len, data_len = addr->ai_addrlen, count = 0, result, i, re;

        if (!(packet_len = construct_packet(data)))
        {
            return 0;
        }
        /*客户端给服务器端发送 NTP 协议数据包*/
        if ((result = sendto(sk, data,
             packet_len, 0, addr->ai_addr, data_len)) < 0)
        {
            perror("sendto");
            return 0;
        }

        /*调用 select()函数，并设定超时时间为 1s*/
        FD_ZERO(&pending_data);
        FD_SET(sk, &pending_data);
        block_time.tv_sec=10;
        block_time.tv_usec=0;
        if (select(sk + 1, &pending_data, NULL, NULL, &block_time) > 0)
```

```
    {
        /*接收服务器端的信息*/
        if ((count = recvfrom(sk, data,
                NTP_PCK_LEN * 8, 0, addr->ai_addr, &data_len)) < 0)
        {
            perror("recvfrom");
            return 0;
        }

        if (protocol == TIME)
        {
            memcpy(&ret_time->transmit_timestamp, data, 4);
            return 1;
        }
        else if (count < NTP_PCK_LEN)
        {
            return 0;
        }

    /*  设置接收 NTP 包的数据结构  */
        ret_time->leap_ver_mode = ntohl(data[0]);
         ret_time->startum = ntohl(data[1]);
        ret_time->poll = ntohl(data[2]);
        ret_time->precision = ntohl(data[3]);
        ret_time->root_delay = ntohl(*(int*)&(data[4]));
        ret_time->root_dispersion = ntohl(*(int*)&(data[8]));
       ret_time->reference_identifier = ntohl(*(int*)&(data[12]));
        ret_time->reference_timestamp.coarse = ntohl *(int*)&(data[16]));
        ret_time->reference_timestamp.fine = ntohl(*(int*)&(data[20]));
        ret_time->originage_timestamp.coarse = ntohl(*(int*)&(data[24]));
        ret_time->originage_timestamp.fine = ntohl(*(int*)&(data[28]));
        ret_time->receive_timestamp.coarse = ntohl(*(int*)&(data[32]));
        ret_time->receive_timestamp.fine = ntohl(*(int*)&(data[36]));
        ret_time->transmit_timestamp.coarse = ntohl(*(int*)&(data[40]));
        ret_time->transmit_timestamp.fine = ntohl(*(int*)&(data[44]));
         return 1;
    } /* end of if select */
    return 0;
}

/*  修改本地时间  */
int set_local_time(struct ntp_packet * pnew_time_packet)
{
```

```
        struct timeval tv;
        tv.tv_sec = pnew_time_packet->transmit_timestamp.coarse - JAN_1970;
        tv.tv_usec = USEC(pnew_time_packet->transmit_timestamp.fine);
        return settimeofday(&tv, NULL);
    }

    int main()
    {
        int sockfd, rc;
        struct addrinfo hints, *res = NULL;
        struct ntp_packet new_time_packet;
        memset(&hints, 0, sizeof(hints));
        hints.ai_family = AF_UNSPEC;
        hints.ai_socktype = SOCK_DGRAM;
        hints.ai_protocol = IPPROTO_UDP;
        /*调用 getaddrinfo()函数，获取地址信息*/
        rc = getaddrinfo(NTP_SERVER_IP, NTP_PORT_STR, &hints, &res);
        if (rc != 0)
        {
            perror("getaddrinfo");
            return 1;
        }
        /* 创建套接字 */
        sockfd = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
        if (sockfd <0 )
        {
            perror("socket");
            return 1;
        }
    /*调用取得 NTP 时间的函数*/
        if (get_ntp_time(sockfd, res, &new_time_packet))
        {
            /*调整本地时间*/
            if (!set_local_time(&new_time_packet))
            {
                printf("NTP client success!\n");
            }
        }
        close(sockfd);
        return 0;
    }
```

附 录 C
（规范性附录）
故障代码表

代码	代码说明	备注
0x0000	电梯无故障	
0x0101	电梯安全回路断路轿厢内有人	
0x0102	电梯安全回路断路轿厢内无人	
0x0201	开门走车轿厢内有人	
0x0202	开门走车轿厢内无人	
0x0301	关门故障轿厢内有人	
0x0302	关门故障轿厢内无人	
0x0400	轿厢在开门区域外停止	
0x1000	运行时间限制器动作	指电梯在规定的时间内没有到达下一楼层的开门区域
0x1100	楼层位置丢失	
0x1301	开门故障轿厢内有人	
0x1302	开门故障轿厢内无人	
0x2511	上极限开关动作轿厢内有人	电梯冲顶
0x2512	上极限开关动作轿厢内无人	电梯冲顶
0x2521	下极限开关动作轿厢内有人	电梯蹲底
0x2522	下极限开关动作轿厢内无人	电梯蹲底
0x2600	电梯运行速度异常	
0x4000	电梯无事件	
0x4200	电梯检修运行模式	
0x4300	消防服务模式	
0x4700	有司机服务模式	
0x6000	自动扶梯和自动人行道无故障	自动扶梯或自动人行道
0x6100	自动扶梯和自动人行道安全回路断路	
0x7000	自动扶梯和自动人行道运行速度异常	
0x7100	自动扶梯和自动人行道运行方向异常	
0x7500	自动扶梯和自动人行道无事件	
0x7700	自动扶梯和自动人行道检修运行模式	
0x9111	电梯停电有人	
0x9112	电梯停电轿厢内无人	
0x9121	电梯启动失败有人	
0x9122	电梯启动失败轿厢内无人	
0x9191	电梯其他故障有人	