设 计 手 册

目 录

[1. 引言 3](#_Toc496191774)

[1.1 概述 3](#_Toc496191775)

[1.2 编写目的 3](#_Toc496191776)

[1.3 参考资料 3](#_Toc496191777)

[2. 架构设计 4](#_Toc496191778)

[2.1 售票机逻辑结构 4](#_Toc496191779)

[2.2 软件架构 5](#_Toc496191780)

[2.3 软件环境 6](#_Toc496191781)

[3. 软件设计 7](#_Toc496191782)

[3.1 软件结构 7](#_Toc496191783)

[3.1.1 有限状态机 7](#_Toc496191784)

[3.1.2 装饰模式 8](#_Toc496191785)

[3.2 软件功能 9](#_Toc496191786)

[3.2.1 业务功能 10](#_Toc496191787)

[3.2.1.1 班次查询 10](#_Toc496191788)

[3.2.1.2 保险预购买 11](#_Toc496191789)

[3.2.1.3 费用支付 12](#_Toc496191790)

[3.2.1.4 票据打印 13](#_Toc496191791)

[3.2.1.5 网购取票 14](#_Toc496191792)

[3.2.1.6 设备模式 14](#_Toc496191793)

[3.2.2 系统功能 15](#_Toc496191794)

[3.2.2.1 参数管理 15](#_Toc496191795)

[3.2.2.2 检测功能 16](#_Toc496191796)

[3.2.2.3 部件检测 16](#_Toc496191797)

[3.2.2.4 软件环境检测 17](#_Toc496191798)

[3.2.2.5 数据维护 18](#_Toc496191799)

[3.2.2.6 时钟同步 18](#_Toc496191800)

[3.2.2.7 广告管理 19](#_Toc496191801)

[3.2.2.8 监控管理 19](#_Toc496191802)

[3.2.2.9 日终结算 20](#_Toc496191803)

[3.2.3 维护管理 20](#_Toc496191804)

[3.2.3.1 车票处理 21](#_Toc496191805)

[3.2.3.2 保单处理 21](#_Toc496191806)

[3.2.3.3 车票查询 21](#_Toc496191807)

[3.2.3.4 系统操作 21](#_Toc496191808)

[3.3 服务功能 21](#_Toc496191809)

[3.3.1 参数管理 21](#_Toc496191810)

[3.3.1.1 参数描述 22](#_Toc496191811)

[3.3.1.2 参数文件管理 24](#_Toc496191812)

[3.3.2 车票处理软件的管理 25](#_Toc496191813)

[3.3.3 车票库存管理 25](#_Toc496191814)

[3.3.4 安全程序 26](#_Toc496191815)

[3.3.4.1 设备识别和确认 26](#_Toc496191816)

[3.3.4.2 访问控制 27](#_Toc496191817)

[3.3.4.3 外部命令 29](#_Toc496191818)

[3.3.4.4 外部通信安全 29](#_Toc496191819)

[3.3.4.5 存储数据的安全 30](#_Toc496191820)

[3.3.4.6 显示器上的显示信息 30](#_Toc496191821)

[3.3.5 机器的驱动和监视 30](#_Toc496191822)

[3.3.6 操作结果 33](#_Toc496191823)

[3.3.6.1 交易和事件记录 33](#_Toc496191824)

[3.3.7 系统管理功能 34](#_Toc496191825)

[3.3.7.1 与车站计算机的通信 34](#_Toc496191826)

[3.3.7.2 票据打印机 35](#_Toc496191827)

[3.3.7.3 显示器命令 35](#_Toc496191828)

[3.3.7.4 操作状态和模式 35](#_Toc496191829)

[3.3.8 运行图 36](#_Toc496191830)

[3.3.9 状态描述 38](#_Toc496191831)

[3.3.9.1 断电状态 38](#_Toc496191832)

[3.3.9.2 正常状态 38](#_Toc496191833)

[3.3.9.3 降级模式 40](#_Toc496191834)

[3.3.9.4 退出服务状态 41](#_Toc496191835)

[3.3.10 瞬时状态 42](#_Toc496191836)

[3.3.10.1 初始化 42](#_Toc496191837)

[3.3.10.2 关闭 42](#_Toc496191838)

[4. 模块设计 42](#_Toc496191839)

[4.1 票务模块（TKB） 42](#_Toc496191840)

[4.2 支付模块（HPay） 44](#_Toc496191841)

[4.3 串行模块接口 45](#_Toc496191842)

[4.3.1 串口通信层 45](#_Toc496191843)

[4.3.2 模块协议层 45](#_Toc496191844)

[4.3.3 模块命令层 46](#_Toc496191845)

[4.3.4 模块应用层 46](#_Toc496191846)

[5. 接口设计 47](#_Toc496191847)

[5.1 班次查询接口 47](#_Toc496191848)

[5.2 预扣座接口 49](#_Toc496191849)

[5.3 出票接口 51](#_Toc496191850)

[5.4 网上订单查询 53](#_Toc496191851)

[5.5 网上订单出票 55](#_Toc496191852)

[5.6 车票操作 57](#_Toc496191853)

[5.7 设备状态 57](#_Toc496191854)

[5.8 保单上传 59](#_Toc496191855)

[5.9 查询当前票号 60](#_Toc496191856)

[5.10 查询客运站售票站点 60](#_Toc496191857)

[6. 数据存储设计 61](#_Toc496191858)

[6.1 购票记录表 62](#_Toc496191859)

[6.2 投保记录表 62](#_Toc496191860)

[6.3 支付日志表 63](#_Toc496191861)

[6.4 异常交易表（凭条） 64](#_Toc496191862)

# 引言

## 概述

此文档是用来描述自动售票机管理软件的结构设计，自动售票机是地铁AFC系统中的一个基本组成部分。

AFC系统旨在通过使用车票的流通来实现对南京地铁公司对运输收入的自动化管理。自动售票机为自动收费系统提供与发售给乘客的运输车票相关的所有服务。这种售票终端为自动化设备，由乘客直接通过身份证获取其订购的车票，人机交互直接由触控显示屏来进行实现的。

## 编写目的

编写车票自动售票机系统软件功能设计书的目的在于从详细设计的角度明确该系统的功能范围，明确与业务主机/终端系统的接口和业务处理模式，使系统开发人员和产品管理人员明确产品功能，可以有针对性的进行系统开发、测试、验收等各方面的工作。

## 参考资料

《BTS设备标准化软件功能规格书-CC-V2.1.docx》

《BTS设备标准化软件功能规格书-TKB-V2.1.docx》

《BTS设备标准化软件功能规格书-TVM-V2.1.docx》

# 架构设计

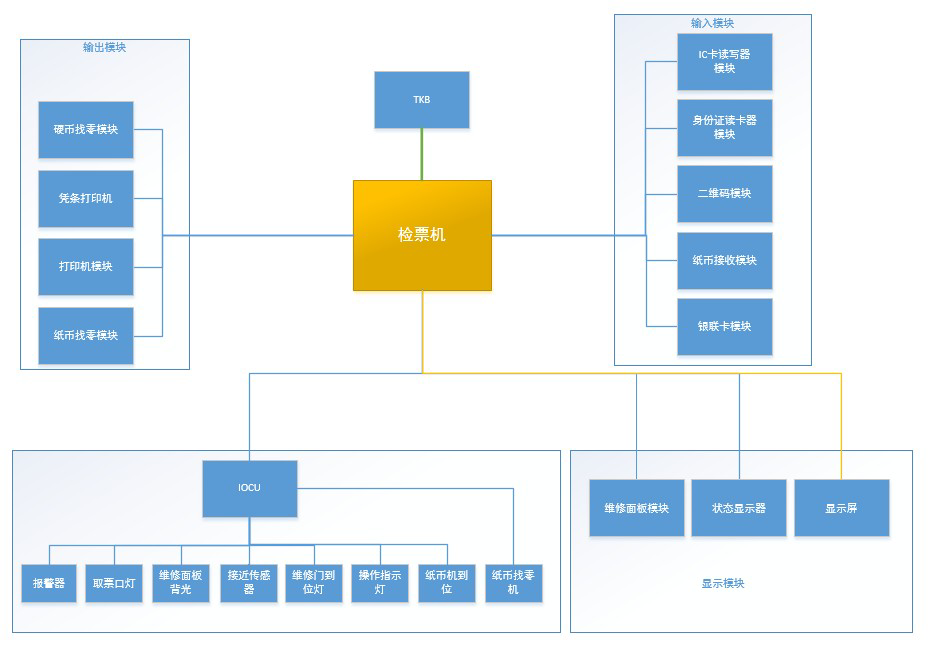
自动售票机采用标准接口方式，与标准票务系统TKB对接，实现不同项目、不同票务系统和不同保险系统的自动售票。

## 售票机逻辑结构

自动售票机的逻辑结构如下：



自动售票机机包含很多的外部硬件模块，主要有：

* 输入模块：主要包括乘车凭证信息输入（身份证读卡器）模块以及支付输入模块（二维码输入、银联卡以及IC卡读写）；
* 输出模块：包括凭条打印机，车票打印机；
* 显示模块：包括用于和乘客交互的状态显示器和显示屏，用于维护人员交互的维修面板模块；
* IOCU模块：IO控制模块，控制包括报警器、取票口灯、维修面板灯光、接近传感器、维修门到位灯、操作指示灯、纸币到位机和纸币找零机等；
* TKB：标准票务系统，TKB作为服务端，对自动售票机提供与第三方票务系统的对接；
* HPay：标准支付系统，HPay作为服务端对自动售票机提供第三方支付接口，HPay按照第三方支付平台的接口、规则接入。

## 软件架构

自动售票应用软件分为四层：接口层，驱动层，逻辑层和应用层。



接口层：实现串行接口，并行接口和网络接口，与设备的协议和指令无关，只实现通用接口功能；

驱动层：根据部件的协议和指令实现部件的驱动功能；

逻辑层：实现部件驱动的逻辑，逻辑层主要实现硬件的逻辑功能，例如提示功能，同时包括了蜂鸣、状态等灯、打印等功能，是对驱动层的封装。

应用层：应用层是售票功能的核心，实现自动售票用户的交互，主要包括售票、取票、维护等应用层的功能。

## 软件环境

自动售票机应用程序运行环境和开发环境：

运行环境：Linux

开发语言：Qt, JAVA

数据存储：SQLite

# 软件设计

## 软件结构

自动售票机应用程序使用Qt编写，利用有限状态机和装饰模式实现售票机的功能。

### 有限状态机

自动售票机的业务功能为有限状态机，包括以下五个状态：班次选择、保单购买、费用支付、票据打印和网购取票。五个状态之间的关系如下：



班次选择：选择合适的班次，包括起始站、日期、数量；

保单购买：根据购票业务配置，可选；

费用支付：使用银行卡、第三方支付平台支付；

票据打印：打印车票或保单；

网购取票：根据网上购票信息取票。

### 装饰模式

根据自动售票机的有限状态和过程，采用装饰模式（Decorator模式，动态地给一个对象增加职责或行为），实现自动售票机的功能。采用装饰模式，不但可以动态的实现对象行为的添加和减少，而且可以方便的调整购票流程，便于灵活配置。

根据自动售票机的3.1.1有限状态机，对于购票过程设计五种个具体的部件（Concrete Component）：

* 班次选择
* 保单购买
* 费用支付
* 票据打印
* 网购取票

对于标准的购票过程（含保单购买），整个过程装饰四次（班次选择，保单购买，费用支付，网购取票）；对于无需保单的购票过程，装饰三次（班次选择，费用支付，网购取票）；对于取票过程，装饰一次（网购取票）。

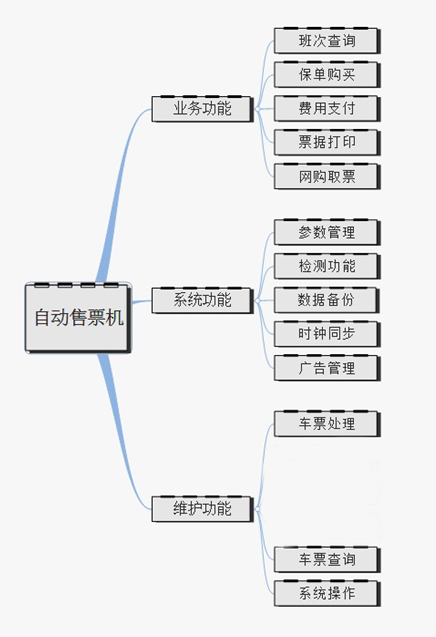
## 软件功能

自动售票机的功能分为三大类：

业务功能：包括售票、取票等为乘客服务的功能；

系统功能：支持系统正常运行的辅助功能；

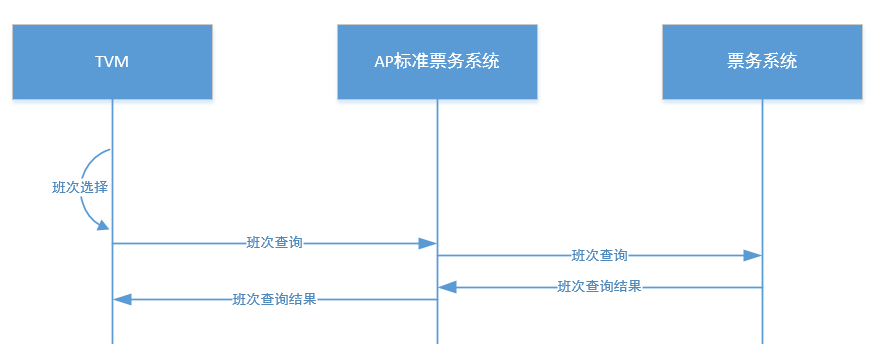
维护功能：为维护人员提供的维护系统的功能。



### 业务功能

#### 班次查询

自动售票机可根据乘客选择的购票条件（如起始站，目的站，日期等组合查询）从票务后台查询出符合条件的车次，供乘客查询与购票。



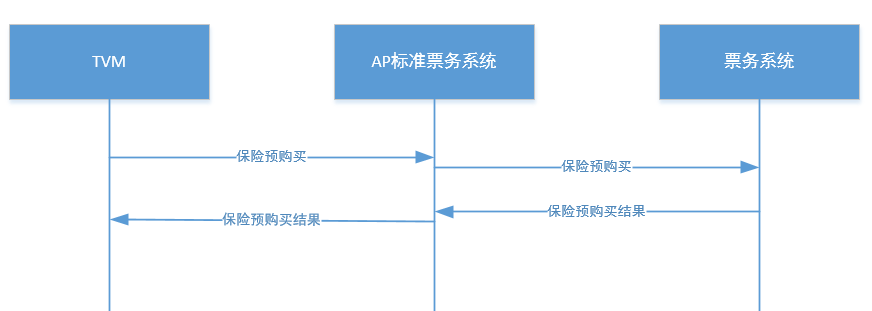
班次查询时，自动售票机将查询条件发送到标准票务系统，标准票务系统将查询条件发送给第三方的票务系统并返回结果。系统返回票务信息供乘客确认。

| **功能名称** | **班次查询** | |
| --- | --- | --- |
| 项目名称 | 项目用途/说明 | 输入方式 |
| 出发城市 | 路径规划时使用 | 乘客输入 |
| 出发站点 | 选择省->市->车站/拼音首字母 | 乘客输入 |
| 出发站名称 | 起始站名称（乘车站） | 乘客输入 |
| 目的城市 | 路径规划时使用 | 乘客输入 |
| 目的站点 | 选择省->市->车站/拼音首字母 | 乘客输入 |
| 目的站名称 | 目的站名称 | 乘客输入 |
| 班次日期 | 选择指定日内的车票（最大15日） | 乘客输入 |
| 班车时间 | 显示当天指定时间之后的班次，默认0点 | 乘客输入 |
| 座位类型 | 座/卧/站（默认全选） | 乘客输入 |
| 运营方式 | 直达/普通（默认全选） | 乘客输入 |
| 班次号 | 指定班次号信息 | 乘客输入 |
| 业务类型 | 自助售票/车站互售 | 系统输入 |
| 请求站 | 查询发生站 | 系统输入 |
| 操作地址 | 请求站所在地址 | 系统输入 |
| 保留字段1 |  |  |
| 保留字段2 |  |  |
| 保留字段3 |  |  |

#### 保险预购买

自动售票机有三种保单购买模式供车站管理人员设置——乘客选择购买保单、强制购买保单、不购买保单。如果配置为不购买保单模式，保险预购买功能不呈现。如果为乘客选择购买保单或强制购买保单，需要首先确认乘客购买数量或读取保险人信息。

保险购买分为两个步骤，第一个步骤是预购买——系统这时能够获取保单价格，但由于乘客还未支付，保险购买的动作还未完成；第二个步骤是支付（和车票支付一起完成），保单的费用支付完成后，整个保险购买才算完成，并取得每个保单的保险号。



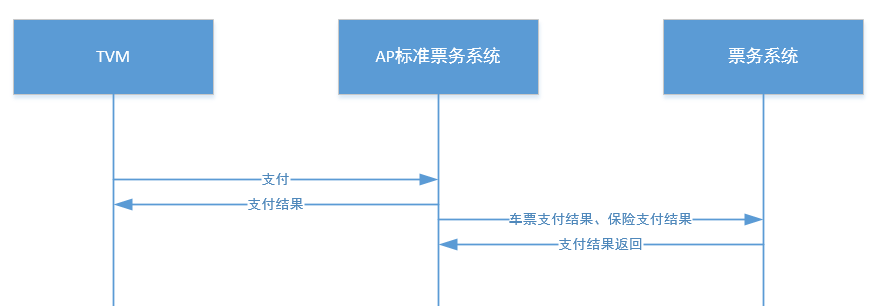
| **功能名称** | 保险出票功能 | |
| --- | --- | --- |
| 项目名称 | 项目用途/说明 | 输入方式 |
| 票务交易号 | 票务交易编号TkbTransID | 系统输入 |
| **投保信息记录体（N）** | 每份保单投保时包含的信息记录体（N） |  |

* 投保信息记录体

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | | 项目用途/说明 | 输出方式 |
| 投保数 | | 投保数量 | 系统输入 |
| **投保信息字段** | | | |
| 座位号 | 锁票座位号 | | 系统输入 |
| 投保人姓名 | 被保险人的姓名 | | 乘客刷证输入 |
| 投保人证件类别 | 身份证/军官证/护照等 | | 乘客刷证输入 |
| 投保人证件号码 | 被保险人的证件号 | | 乘客刷证输入 |
| 投保人性别 | 被保险人的性别 | | 乘客输入 |
| 投保人出生日期 | 被保险人的出生日期 | | 乘客输入 |
| 受益人姓名 | 保单打印受益人的姓名 | | 乘客刷证输入 |
| 受益人证件类别 | 身份证/军官证/护照等 | | 乘客刷证输入 |
| 受益人证件号码 | 保单打印受益人的证件号 | | 乘客刷证输入 |

#### 费用支付

费用包括车票费用和保单费用。如果费用支付失败，需要确认保险购买失败，以实现交易回滚。



##### 银联支付

自动售票机通过第三方银行卡终端（POS机）获取支付卡号等信息并实现支付功能，并将支付结果返回给自动售票机。

第三方银行卡终端采用串口通信方式，银行POS起到了取卡信息与密码输入的作用，除了取卡信息和密码输入外，还需要控制POS机与收银机之间的交互，以便多批不同数据的处理。

##### 其他支付

其他支付模块是指微信、支付宝等在线支付方式，通过手机扫描TVM二维码方式实现在线快速支付。

手机APP（刷身份信息加密后产生的二维码）离线支付【待定】。

#### 票据打印

当费用支付成功后，进入车票打印界面，车票打印成功后返回主界面。

##### 打印车票

支付成功后，调用车票打印机打印车票。提示用户打印车票进度，有界面提示和语音提示两种。

##### 打印保险单

支付成功后，调用保单打印机打印保单信息。提示用户打印保单进度，有界面提示和语音提示两种。

#### 网购取票

##### 网上订单查询

通过互联网或其他方式在网络上预定车票后，可点击“网上购票”按钮进入取票界面。

在网上取票界面，通过身份证读卡器或直接输入的方式获取身份证号，验证正确后显示车票信息。

##### 网上订单出票

网上预定车票确认无误后可通过打印出票。

#### 设备模式

* 保单不足（含本机和后台返回）——如果强制购买保单，进入暂停服务；其他提示用户，进入降级模式；
* 身份证读卡器异常——如果实名或者强制购买保单，进入暂停服务；其他提示用户，进入降级模式。
* 车票打印机异常——进入暂停模式；
* 凭条打印机异常——进入暂停模式；
* 二维码扫描枪（取票专用）异常——进入降级模式；
* 银联POS机异常——进入不支持银联支付模式；

### 系统功能

系统功能包括如下内容：

* 参数管理：对程序运行的参数设置，包括部件连接串口、波特率、名称等；
* 检测功能：对程序运行环境检测，包括部件检测和软件环境检测；
* 数据备份：对程序运行过程的交易数据和日志备份；
* 时钟同步：同步自动售票机的时间；
* 广告管理：对自动售票机上展示的广告时间、次数等进行管理。
* 日终结算：与第三方票务系统对账、结算。

#### 参数管理

参数管理是对自动售票机上运行的应用软件参数设置，包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 部件名称 | 检测方式 | 备注 |
| 1 | 程序参数 | 安装部件类型、串口号、波特率等 |  |
| 2 | 各部件通信超时时间长度 |  |
| 3 | 标准票务系统的IP地址，端口号 |  |
| 4 | 设备登录密码、认证密码 |  |
| 5 | 交易保存天数 | 过期交易自动清理 |
| 6 | 日志保存天数 | 过期日志自动清理 |
| 5 | 运营参数 | 银联签到时间 |  |
| 7 | 运营时间 |  |
| 8 | 车站号、车站名称、操作员信息 |  |
| 9 | 提前购票天数 |  |
| 10 | 默认的起始站 |  |
| 11 | 凭条打印机标题和广告 |  |
| 12 | 功能超时时长 |  |
| 13 | 开门超时时间 |  |
| 14 | 支持的支付方式 |  |
| 15 | 售卖车票种类 | 1全票，2学生票，4童票 |
| 16 | 购买保险方式 | 0无保险，1自动投保，2用户手选 |
| 17 | 取票方式 | 1身份证，2身份证+密码，4取票号+密码，8手机号+密码，16二维码+密码 |
| 18 | 每次最多购买的车票张数 |  |
| 19 | 应用程序 | 应用程序、固件、资源文件等 |  |

#### 检测功能

检测功能主要针对维护人员，对自动售票机的部件和软件环境进行检测，以查看是否是否满足运行条件。

#### 部件检测

部件检测功能对自动售票机上的部件逐一检测，反馈给系统维护人员，主要检测内容包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 部件名称 | 检测方式 | 备注 |
| 1 | 车票打印机 | 串口通信，打印样张 |  |
| 2 | 保单打印机 | 串口通信，打印样张 |  |
| 3 | 凭条打印机 | 串口通信，打印样张 |  |
| 4 | 碳带剩余 | 热转印打印机，通过打印车票数量统计，阈值可配置 |  |
| 5 | 扫描枪 | 串口通信，扫描二维码 |  |
| 6 | 银联卡外设 | 串口通信，刷样卡 |  |
| 7 | 身份证读卡器 | 串口通信，读身份证 |  |
| 8 | 触摸屏 | 通过四角定位，确认并调整 |  |
| 9 | 顶灯 | 闪烁，人工确认 |  |
| 10 | 蜂鸣器 | 蜂鸣，人工确认 |  |
| 11 | 取票灯 | 闪烁，人工确认 |  |
| 12 | 维修面板背光 | 闪烁，人工确认 |  |
| 13 | 接近传感器 | 靠近，人工确认 |  |
| 14 | 操作指示灯 | 闪烁，人工确认 |  |

#### 软件环境检测

软件环境检测功能是对自动售票软件运行的环境检测，包括操作系统版本、组件、本地数据库、网络接口进行检测。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 检测项 | 检测方式 | 备注 |
| 1 | 操作系统版本 | 检测操作系统内核版本是否高于软件所需最低要求 |  |
| 2 | 必须组件检测 | 必须组件以及其版本检测 |  |
| 3 | 本地数据库 | 本地数据库文件存在、结构检测 |  |
| 4 | 网络接口 | 是否可以连接到站务系统 |  |
| 5 | 票务后台 | 是否可以连接到三方票务平台 |  |
| 6 | 保险后台 | 是否可以连接到保险平台 |  |
| 7 | 支付宝 | 是否可以连接到支付宝平台 |  |
| 8 | 微信 | 是否可以连接到微信支付平台 |  |

#### 数据维护

售票交易数据、日志信息存储在本地的SQLite数据库内，按照一定周期（每隔多少天）清理SQLite内的历史数据。

实现方式：

* 程序启动后判断是否需要执行数据维护（超过指定天数）执行数据维护；
* 检索指定日期之前的交易数据和日志数据，以文本形式存储在本地log目录下；
* 删除指定日期之前的交易数据和日志数据；
* 调用VACUUM命令释放磁盘空间。

#### 时钟同步

自动售票机的时钟由后台服务器的时钟来校正。

自动售票机通过报文中的后台服务器时间来校正本地的时钟，保持本地时钟与后台服务器的时钟一致。

#### 广告管理

广告管理是指对缺省主界面上展示的广告（图片、视频）播放次数、播放时长进行统计，并按照设置的要求切换、终止。

#### 监控管理

对设备主机和外设进行统一监视和控制。

##### 状态监视

支持主机信息监视，如CPU占用、内存使用、文件系统占用、硬盘IO性能等状态监视，出现异常或故障时，及时报警。

支持外设信息监视，如二维码扫描枪、身份证读卡器、IC卡读卡器、通道等。

##### 设备控制

设备接收CC端发送八种设备命令：

* 停止服务
* 开始服务
* 上传数据，主机信息、外设信息等
* 上传日志，即Log文件
* 时钟强制同步
* 下载参数
* 设置TVM类型：售票机/查票机/取票机
* 运营结算，结算清单参考“日终结算”

#### 日终结算

运营结束后，生成结算账单，并支持小票打印输出。包括设备操作员工号、结算单号、运营开始结束时间段、车票使用号段、保单使用号段、售票张数/金额、保单张数/金额、银联收费笔数/金额、互联网支付类型/笔数/金额、取票张数/金额、实收总金额、未出票失败多收金额、未出票失败多收笔数、已出票失败多收金额、已出票失败多收笔数等。

### 维护管理

维护功能主要针对运营人员操作功能，包括：

* 车票处理：输入打印机上空票的起始和结束条码；
* 保单处理：输入打印机上空保单的起始和结束条码；
* 碳带处理：查询碳带使用情况，提示是否需要更换；
* 车票查询：查询到自动售票机已出售的最近车票信息；
* 系统操作：操作自动售票机的系统功能。

#### 车票处理

车票操作主要是输入打印机上空票的起始和结束条码。

#### 保单处理

保单操作主要是输入打印机上空保单的起始和结束条码。

#### 车票查询

车票查询可以查询到本台售票机已出售的最近10张车票信息。

#### 系统操作

系统操作主要是操作TVM的系统，包括设备重启、设备状态、设备复位、设备开机、程序退出等。

## 服务功能

### 参数管理

为了实现最大程度的灵活性，可通过下载到自动售票机中的参数和具体的车票处理软件（商业规则）来对自动售票机操作进行配置和执行。

这些参数中的大部分都是通过外部通信接口作为不同的文件传输的。这些参数在中央控制室级别上被管理和更新。乘客操作员并不参与参数处理。在自动售票机级别上，这些程序自动执行。只有维护操作员可以通过用于维护的膝上型计算机访问这些程序。

#### 参数描述

有两类自动售票机参数：配置参数、设备操作参数（EOD）。

配置参数：

有些参数不常修改，它们一般保存在机器的常驻内存中。每次当自动售票机与中央计算机连接时，它们就被下载到自动售票机中校验自动售票机参数并在发生故障时对这些参数进行重新初始化。这些参数包括：

（1）生产数据

设备所有者ID

制造商序列号

（2）本地化数据

服务供应商ID

车站ID

线路ID

（3）安装数据

以太网配置：网络机器的名称、协议类型、IP地址、端口编号、文件传输协议登陆参考。

文件管理：参数文件路径，软件文件路径，操作结果文件路径。

（4）设备操作数据

根据外部通信的类型，在每次进行自动售票机连接时，每隔一段时间（如果可能的话），当前日期和时间就被送入作为个人计算机时钟。当不存在可用的外部通信时，检票员或维护操作员可通过人工程序对这些数据进行更新。其它的设备操作参数分别归属于不同的文件，通过外部通信连接（站台计算机连接或膝上型维护工具）来下载。

基本的参数文件包括：

（1）参数参考文件（主配置文件）

该文件列出了不同的版本和文件适用日期，这些信息必须存在于自动售票机内存中。该文件必须首先下载。通过这个文件，自动售票机能够发现任何丢失的或错误的参数文件。

（2）共用参数文件

共用参数文件包括以下数据：

• 车站列表：自动收费系统中的车站（名称和编号）列表。

• 通用系统参数：“核算日”、“旅行车票最大有效期”的定义，票据打印的配置等。

• 费用：该文件包括在自动收费系统中的各车站之间进行的一次旅程的费用。

• 车票面值折扣：适用于不同车票类型的折扣，用来计算可折扣车票的旅程费用，其值以百分比表示。

• 罚款：该文件包括每项费用的价值以及对不同的冲突进行调节所收取的额外费用。

* 费用产品列表：该文件包括对每种自动收费系统中的车票类型的描述。

#### 参数文件管理

每个参数文件都有一个独立的版本编号，包括在其文件名称中。每当发生修改时，相关文件的版本编号就会增加。每当准备使用一个新的参数文件版本时，中央控制室会更新参数参考文件并建议使用新版本。通过这些文件接收到的所有数据均存储在自动售票机的存储器中。

系统可以同时管理两个版本：当前（或现行）版本以及将来版本。每个文件的每个版本均附有应用日期。系统中使用的所有参数文件都是于有效日期之前被下载到可用设备中。此后它们就将一直保存在机器的存储器中直到有效日期到期。在有效日期到来的那天（或在此之后），每台自动售票机将把未来参数版本转变为当前操作版本，而原先的现行版本将被删除。

此时，旧的版本将丢失。不可能再回到此版本并将其作为当前版本，只有通过用一个比当前日期更小的有效日期和一个比当前版本编号大的版本编号重新下载后才能实现这一点。

当在“核算日”下载了这样一种版本时，如果没有交易在运行，那么它将立即变成现行版本。如果有交易正在运行，那么这种改变将在交易完成后发生。如果一个未来版本被下载到自动售票机中，而未来版本已经存在，那么该版本将被新的未来版本所替代。如果自动售票机探测到了参数错误，那么参数文件必须重新载入。

### 车票处理软件的管理

新版本的车票处理软件应以一个应用日期和一个版本编号事先进行下载。在有效日期内，营业日开始的那天，新的版本将确定。此时，自动结束轮班、系统的关闭和系统的重新初始化将自动进行。如果在下载新的现行版本时没有轮班处于打开状态，那么新的现行版本将立即生效，此时自动售票机应关闭。使用自动售票机配置功能可以在人机界面上看到现行和未来设备操作参数和软件的版本。

### 车票库存管理

当一批新的非接触式智能卡被送入到售票区的自动售票机中时，系统将启动一个进程，操作员根据交货报告上指明的信息，输入卡的数量，同时设备中的库存情况将被更新，新的库存值被生成，可以打印收据（根据需要）；这个过程将被纪录在事件日志中并被作为消息发往中央系统。

当一个新的票箱被插入到筹码送票器中时，操作人员输入票箱的号码（该号码被标注在票箱上），同时向中央发送一个请求，在中央系统的回复中将指明此票箱所容纳的筹码数量，此数量值将被用来更新车票库存的值；当与中央的通讯链路无效时，票箱中筹码数量信息将在来自车站的报告中指明，然后该数量将由操作员输入。每当新的车票库存修改（售票）后，库存值将减少。当一张车票被退还或取消时，退还／取消的车票库存将自动更新。所退还的车票将返还给中央库存。当一张车票不能用非接触式读／写模块进行读或写操作时，故障车票库存将自动更新。故障车票将返还给中央库存进行销毁。这两种库存值在每次轮班结束后将被重置。

筹码库存被划分为两个库存，筹码自动填充机中的每个票箱为一个库存。每当相应的库存值为零时，票箱初始库存便可根据操作员的要求进行设置（通过库存调整命令）。每当弹出一个筹码后，所计算出的票箱库存将减一。操作员可要求进行清点（通过清点命令）。清点意味着清空票箱以便对筹码进行计数。一个专门的交易信息含有计算出的库存，而自动售票机理论上的库存将被传送到中央计算机。新库存为零。

在更换当前票箱前必须将其清空。这可以应操作员的要求来进行（通过清点命令）。新的票箱被固定在分发器上并且将为其设置一个初始票箱库存（通过库存调整命令）。然后按照以上所述方法对库存进行管理。

### 安全程序

#### 设备识别和确认

每台机器均配有一个安全模块。该模块为Schlumberger公司生产的Cryptoflex智能卡，该卡是设计用于保证应用的安全的。该卡所使用的技术是基于ISO7816-1-2-3-4安全标准。由中央计算机将设备编号、机器类型和公共密钥载入智能卡中。然后在机器中对该卡进行设置。此时，安全模块允许系统对机器进行识别和认证以及实现不同的用于传输的安全进程。

#### 访问控制

##### a：操作对话，登陆程序

在操作员能够访问自动售票机命令之前，他必须通过一个身份识别程序登陆。只有在被识别为经授权可操作自动售票机的操作员之后，他才可以使用所有的命令（交易、维护、服务）。

为实现此目标，每个操作员分配有一个分级代码。自动售票机可识别的不同的代码包括：

乘客操作员代码；

检票操作员代码；

维护操作员代码。

每个分级代码定义一套可访问功能。当识别程序正确运行后，操作员即成功登陆。根据他自己的分级代码级别，机器自动以相应轮班类型打开轮班：销售进程，检票员进程，或维护进程。

##### b：识别程序

当登陆过程被批准时（正常状态），分配给识别程序的屏幕就会显示。首先，操作员必须输入他的员工编号和他的个人身份号码。然后机器会检查该员工编号是否存在于员工白名单中。如果该员工编号不在白名单中，那么登陆就会被拒绝。如果员工编号存在于白名单并且是有效的，那么与该编号相关的一个参数（在列表中）将指出是否需要检查该员工的通行证。如果需要，那么该员工的通行证必须提供给通行证读卡机以读取通行证数据。然后再进行其它检查：

输入的代码必须与从通行证（如果需要的话）中读取的代码一致；

通行证的有效期必须没有过期；

员工通行证不能出现在黑名单中；

在通行证（如果需要的话）上所读取的分级代码应当与白名单上的分级代码相同，从而授权该员工可以访问半自动售票机。如果不要求进行通行证读取，那么上述检查将通过使用存储在白名单中的相应数据来进行。

如果通行证模块出现故障（而又需要有通行证），那么该员工的通行证不能读取。只有那些在员工列表中享有权利的员工才可以登陆。建议仅将此项权利留给检票员或维护操作员。

##### c：退出程序

自动售票机退出程序是通过按压键盘上的一个退出按钮实现的。

在一次轮班期间，自动售票机记录所有的交易。这些记录连接于此时已登陆的操作员。这样自动售票机便可以为每个操作员轮班产生核算和现金对帐报告。

自动售票机在每个轮班结束（退出程序）时产生一份报告。它概括出所进行的交易、商业轮班中自动售票机的车票库存或者维护轮班中的维护次数。如有需要，可在一般参数中对此项功能进行设置以禁止此项功能。

在关闭某个轮班并打印报告前，自动售票机会要求操作员输入收入总额，这个收入总额将打印在报告上并传送给车站计算机。如果此时存在外部通信连接，当操作员退出时，结束轮班报告将被传送给中央控制室。然后结束轮班报告将自动在票据打印机上打印出来（如果经某个设备参数授权的话）。

#### 外部命令

一个外部的“服务中／退出服务”命令可以被传送给自动售票机。退出服务命令禁止操作员打开一次轮班。该命令并不会自动关闭已打开的轮班，但会用一条警告信息来建议操作员“通过操作关闭”，这样商业功能就变得无效了。该命令只能由一位检票员进行本地修改。

其它的命令包括：

自动售票机状态请求；

特殊费用处理模式；

自动售票机模式；

故障模式。

#### 外部通信安全

通信安全主要是通过每次传输中的校验和来实现的，因为这样便可以保证在传输过程中通信没有被偶然地或故意地破坏。

#### 存储数据的安全

所有的敏感数据（操作结果）都存储在数据库中。只有那些应用程序才可以出于本身使用的目的被授权访问数据库。

#### 显示器上的显示信息

在进行交易时，自动售票机将把所有必要的信息显示在显示器上给乘客看。信息将以英文或中文显示。机器中配置有一种默认语言（汉语）。在交易时通过使用一个专门的键可以对此选项进行修改。当制作一个筹码或新卡时，机器将使用默认语言，直到乘客选择了一个新的选项为止。在每次交易结束后，机器将重新回到默认语言。

乘客信息包括：

* 欢迎信息；
* 车票种类；
* 车票出售时的价值。

### 机器的驱动和监视

某些程序是由自动售票机的处理器在每个“核算日”结束后自发启动的：

清除旧的交易和事件记录（超过7天或70000宗以上的交易）；

检查存储的数据和数据库；

检查参数版本应用日期和执行新的版本；

检查车票处理软件版本的应用日期和执行新的版本；

将参数状况（现有文件列表，默认值等）上传到车站计算机／中央计算机；

自动结束轮班（如果没有关闭，并且如果需要改变软件版本的话）。

如果此时某个操作员仍在工作，那么将会显示一个警告信息，并且轮班的自动结束将被推迟到正在进行的交易结束之后。接下来在轮班结束后将开始其它程序。

故障探测和相应的处理是一种刚性进程。由自动售票机处理的各种故障包括：

（1）外部通信故障；

在出现通信故障期间出现的事件和交易将在机器中存储7天时间，然后机器将自动置为退出服务。机器中共可存储70000宗以上的交易。在通信重新恢复后所保存的数据将被传送至车站计算机／中央计算机。同时，自动售票机将把参数状态传送至车站计算机／中央计算机控制室。

（2）电源故障；

车站计算机／中央计算机配有备用电源，可提供６０分钟的独立电源和一个关闭信号（通过通信连接来传输的命令）以完成最后一次交易，当独立电源电力不足时机器将自动关闭。不能停止一个关闭进程，因为该进程允许：

完成一宗进行中的交易；

关闭正在运行的轮班；

更新和关闭不同的文件和数据库；

纠正每个内部任务的中断。

在下一次重新接上电源时，自动售票机将进行初始化。

（3）车票制作模块故障；

当一宗交易正在进行时，如果探测到故障，那么在对卡进行编码前交易将被取消。如果故障是发生在对卡进行编码的过程中，那么机器将记录下事件中卡的编号（如果知道的话）和交易参考。

（4）打印机故障；

如果自动打印选项（参数）被赋予某种功能，那么这些功能将是可用的。在其它情况下，这一点对运行中的轮班没有什么具体影响。在交易结束时必须告知用户他无法得到收据。与操作系统或数据库有关的错误在可能的情况下将由其自身进行管理，通常无需应用程序加以干涉。

（5）参数和车票处理故障。

自动售票机不能执行任何程序，机器退出服务。

（6）认证故障；

车站计算机中的认证过程在每次进行重新连接后执行，它是基于时间基础上的。当发生故障时，只要非接触式智能卡／非接触式智能筹码读写器没有问题，自动售票机将处于降级模式。

（7）筹码填充机故障，或票箱为空。

票箱将不能吐出筹码（对相应的票箱而言）。

在上述每种情况下，事件（以及相反事件）信息将被显示并且机器将发出声音信号来警告操作员。根据不同的事件和来自车站计算机的命令，自动售票机自动定义操作模式。该进程为刚性进程。如果故障将自动售票机的主进程（报警）锁定，那么所产生的模式可能会失效。否则的话，自动售票机可能会以降级模式运行。

### 操作结果

#### 交易和事件记录

每宗交易都将在数据库中产生一个交易记录，该交易记录也将传送至车站计算机。这些交易将保存在一个临时文件中，每隔一段时间或者当交易达到一个预先定义的数目时该文件就会传送到车站计算机中。

自动售票机还将记录下当时登陆到机器中的操作员进行的所有交易。每宗交易以一个递增的数字编号。该编号从不重置。交易记录的主要内容包括：

日期和时间；

交易类型；

车票类型；

交易被拒绝的原因或交易的数额。

在每次轮班结束时自动售票机将产生一份报告，显示出销售值和车票库存使用情况。这样每位操作员在每次轮班结束时就能够对售票和库存进行结算。如果需要的话，可通过对配置文件进行设置来禁止此项功能。

根据交易类型和车票类型来划分的两组交易计数器（概要或审计）在每次交易结束后将进行更新。一组是在每次轮班开始时重置，另外一组是在每个营业日结束时重置。

关于自动售票机活动的每个事件和故障（以及相反的事件）都将记录在数据库中并将被实时地传送到车站计算机中。

这些事件包括：

轮班开始／轮班结束；

暂停／暂停结束；

库存变化；

参数或软件版本的变化；

操作模式的变化；

接收到外部命令；

所有以上所述的故障。

### 系统管理功能

#### 与车站计算机的通信

与车站计算机的通信可以传输：

参数文件；

临时交易文件；

充值；

售票；

退票；

对非接触式智能卡的调整；

被取消的交易；

被拒绝的交易；

库存事件；

在线事件信息；

事件列表中的所有信息；

来自车站计算机的命令；

概述请求；

交易和事件记录请求；

机器状态请求；

模式命令；

自动售票机请求；

退票授权；

换票授权；

票箱容量信息。

#### 票据打印机

票据打印机通过一个串行驱动器进行管理，其状态定期得到检查。

#### 显示器命令

显示器通过一个串行驱动器来管理。该设备并不返回任何状态。

#### 操作状态和模式

自动售票机的“正常状态”是指：

* 没有探测到电源故障；
* 没有出现“报警”。

当处于正常状态时，自动售票机能够以下述模式运行：

1. 商业模式：商业轮班正在运行中；
2. 维护模式：维护轮班正在运行中；
3. 检票员模式：检票员轮班正在运行中；
4. 系统命令锁定模式（退出服务）。

在每种模式下，功能菜单只提供那些为特定的配置所授权的功能。

当正常状态的条件之一没有满足时，自动售票机将以降级模式运行或转变为“出错状态”或“断电状态”。到底处于哪种状态将取决于所出现的不正常情况的严重程度。

瞬时状态包括：

初始化：在电源开关打开时；

关闭：在电源开关断开，更换软件，或不间断电源发生故障时。

### 运行图

如何理解运行图

**状态１**

**瞬进状态 1**

**事件１**

**事件2**

**事件3**

**状态 2**

在同一状态在只有一种模式起作用

正常状态

报警结束

登陆

退出

报警

报警

初始化结束

电源开关打开

出错状态

关闭

初始化

电源开关断开

因电源故障而处于等待状态

关闭结束

主电源故障信息

### 状态描述

#### 断电状态

机器中所有的部件均为无源的。数据保存在个人计算机磁盘、操作系统寄存器、安全模块和外围设备（如果存在的话）的内存中。

#### 正常状态

##### 待机模式

当没有操作员操作时，机器将运行在此状态下。此时机器正在等待一个身份识别程序的开始。可用的功能包括：

对机器的驱动和监视；

外部通信管理；

参数和软件管理；

运行结果管理；

执行外部命令。

##### 商业模式

当有乘客操作员操作时，自动售票机将运行在此模式下。可用功能包括：

对机器的驱动和监视；

外部通信管理；

向车站站控制室传输参数、交易、事件和记录文件；

接收和处理系统命令；

对参数和车票数据处理的管理；

参数和软件的存储；

新版本的具体执行（直接安装）；

乘客服务；

售票和充值；

冲突调节；

交易和事件日志；

库存管理

##### 维护模式

自动售票机按要求提供有诊断软件，用来测试机器的基本操作和连接的外围设备。服务人员可使用这些程序来诊断小至线路可代替部件级别的问题。可用功能包括：

对机器的驱动和监视；

外部通信管理；

接收和处理系统命令；

测试车票上的商业功能；

对参数和车票数据处理的管理；

事件日志；

执行维护程序；

退出程序。

#### 降级模式

引起降级模式的原因包括：

* 票据打印机故障：这一事件将被显示和记录，但并不涉及模式的改变。通过有效的功能，交易将继续进行，但乘客将得不到收据。
* 外部通信故障：这一事件将被显示和记录，但并不涉及模式的改变。被禁止的功能只是那些涉及到与中央计算机进行在线通信的功能，如：退票、换票等。所有可能进行的交易都将被存储7天或存储的交易达到70000宗，然后机器将转入出错模式。
* 车票制作故障：该事件将被显示和记录。在这种情况下，可登陆到维护模式。

#### 退出服务状态

当存在以下两种情况时自动售票机将运行在该模式下：

存在锁定报警，因此无法进行交易；

接收到了系统被锁定的命令（自中央计算机或本地计算机）。

取决于引起退出服务的原因，可用的功能包括：

对机器的驱动和监视；

外部通信；

传输参数、交易、事件和日志文件；

接收和处理系统命令；

对参数和车票数据处理的管理；

参数和软件的存储；

执行新的版本；

事件日志；

登陆维护模式。

### 瞬时状态

#### 初始化

该模式联系于初始化程序，包括：

外部通信；

初始化和对外围设备的检查；

对数据和文件完整性的检查；

启动应用程序；

下载参数和软件；

载入运行结果（如果有要求的话）；

运行模式的初始化。

#### 关闭

在此状态下正在运行中的交易将被完成，然后机器将停止于一个正确的状态（结束任务等）。在结束关闭状态后，模式将通过命令被锁定。

# 模块设计

## 票务模块（TKB）

提供RESTful API供设备层系统调用。标准票务系统作为服务端，设备作为客户端。客运站第三方票务系统、保险平台由其他开发商提供，标准票务系统按照其指定接口、规则接入。

TKB调用CC层的RESTful API接口，将每笔票务数据上传至CC服务层。

第三方站务系统

第三方票务系统

保险系统

CC系统

售检票设备应用

外设模块

票务模块

通讯模块

数据模块

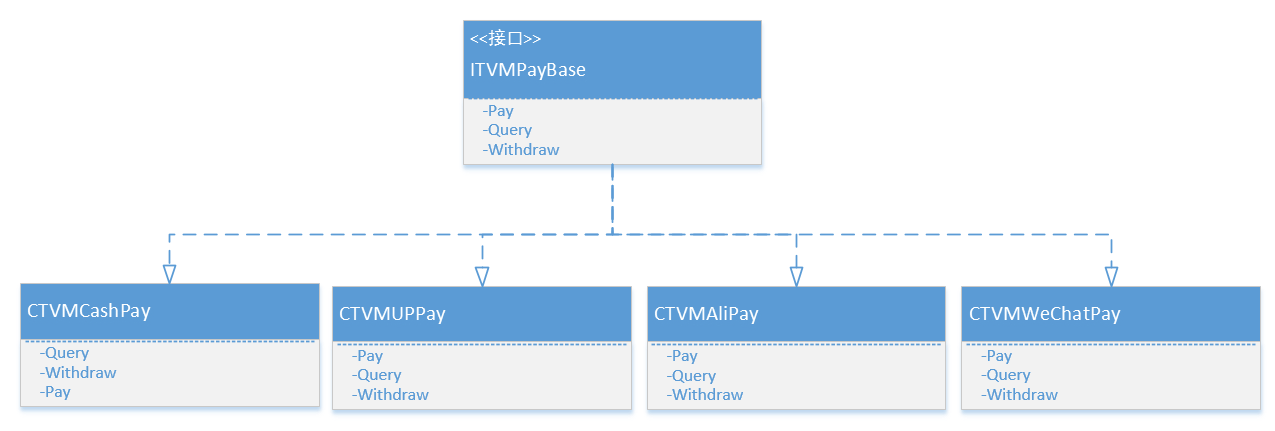
应用内核

标准票务服务TKB

## 支付模块（HPay）

提供RESTful API供设备层系统调用。标准支付系统（以后简称HPay）作为服务端，设备作为客户端。HPay按照第三方支付平台的接口、规则接入。HPay调用CC层的RESTful API接口，将每笔支付交易数据上传至CC服务层。

支付接口采用接口——实体类的方式来实现，编写符合所有支付特征的接口——ITVMPayBase，具体的支付类如银联支付（CTVMUPPay）、支付宝（CTVMAliPay）和微信（CTVMeChatPay），实现接口ITVMPayBase。



## 串行模块接口

二维码扫描枪（取票专用），凭条打印模块、打印机模块、状态显示器、纸币找零模块、纸币接收模块、硬币找零模块、维修面板模块、身份证读卡器模块以及银联卡支付模块，包括可能增加的IC卡读写器模块，均采用RS232接口通信，所有这些模块采用相同的驱动模式。



### 串口通信层

串行通行层负责串行通信实现，根据不同模块获取的输入信息，向主控制模块发送经协议层解析后的数据控制信息。

### 模块协议层

每个串行模块采用不同的协议模式，具有各自的串行通信报文头、报文尾、校验模式等，协议层负责每个模块协议的实现。

封装好的协议层符合以下接口：

* 发送数据：封装发送给IOCU的数据；
* 接收数据：接收IOCU返回的带协议的数据，并且解析；
* 发送接收数据：封装前两个函数。

### 模块命令层

模块命令层实现每个模块的命令，是根据模块接口文档中命令的定义来实现，每个命令实现一个接口。

### 模块应用层

模块应用层是根据业务需要对模块的命令封装，有可能是一个命令，也有可能是多条命令组合而成。

# 接口设计

## 班次查询接口

接口方法[List<HashMap<String, Object>> queryAtmBusData(String fields1, String fields2,String fields3)]

1. 说明

只能查询到发车日期在当前系统时间+10分钟之后的票价不为0、有余票数且为普通班的班次。

**需票务系统提供“省 - 市 - 客运站”三层的列表文件给自动售票机，售票机直接回传给票务后台以获取班次。**

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 表示发车日期，格式为20060303。 |
| Fields2 | 表示上客运站名称。 |
| Fields3 | 表示到站名字(汉字)，如桂林。 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** | |
| --- | --- | --- |
| **Object[0]** (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） | |
| Fields2 | 返回信息描述 | |
| **Object[1]** | | |
|  | Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 发车日期，格式为20120220 |
| Fields2 | 车次 |
| Fields3 | 发车时间，格式为1332，即13点32分发车。 |
| Fields4 | 上客运站，中文 |
| Fields5 | 目的地，中文 |
| Fields6 | 终点站，中文 |
| Fields7 | 途时 |
| Fields8 | 客座数 |
| Fields9 | 车型 |
| Fields10 | 余票 |
| Fields11 | 上客运站代码（客户端调用下文接口订票时传回） |
| Fields12 | 目的地代码（客户端调用下文接口订票时传回） |
| Fields13 | 始发站代码（客户端调用下文接口订票时传回） |
| Fields14 | 全票价，服务器实际返回的价格 |
| Fields15 | 童票价，服务器实际返回的价格 |
| Fields16 | 燃油费，服务器实际返回的价格 |
| Fields17 | 站务费，服务器实际返回的价格 |
| Fields18 | 全票合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields19 | 童票合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields20 | 目前可携带的免票儿童数量 如：01 20160301新增 |
| Fields21 | 是否流水班1为流水班 20160301新增 |
| Fields22 | 线路名 20160301新增 |
| Fields23 | 是否实名售票 20160628新增 |
| Object[1] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 发车日期，格式为20120220 |
| Fields2 | 车次 |
| Fields3 | 发车时间，格式为1332，即13点32分发车。 |
| Fields4 | 上客运站，中文 |
| Fields5 | 目的地，中文 |
| Fields6 | 终点站，中文 |
| Fields7 | 途时 |
| Fields8 | 客座数 |
| Fields9 | 车型 |
| Fields10 | 余票 |
| Fields11 | 上客运站代码（客户端调用下文接口订票时传回） |
| Fields12 | 目的地代码（客户端调用下文接口订票时传回） |
| Fields13 | 始发站代码（客户端调用下文接口订票时传回） |
| Fields14 | 全票价，服务器实际返回的价格 |
| Fields15 | 童票价，服务器实际返回的价格 |
| Fields16 | 燃油费，服务器实际返回的价格 |
| Fields17 | 站务费，服务器实际返回的价格 |
| Fields18 | 全票合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields19 | 童票合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields20 | 目前可携带的免票儿童数量 如：01 20160301新增 |
| Fields21 | 是否流水班1为流水班 20160301新增 |
| Fields22 | 线路名 20160301新增 |
| Fields23 | 是否实名售票 20160628新增 |
| …… | |
| Object[n] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| …… | |

1. 调用存储过程

名称:AutoSale\_QueryAtmBusData\_NEW

参数:@ClassDate @ StationName @ ArriveName

## 预扣座接口

接口方法[List<HashMap<String, Object>> reserveAtmTickets（String fields1,String fields2, String fields3,String fields4,String fields5,String fields6,String fields7, String fields8,String fields9)]

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 表示始发站代码，如001 |
| Fields2 | 表示发车日期，格式为20120220。 |
| Fields3 | 表示车次。 |
| Fields4 | 表示上客运站代码，如001 |
| Fields5 | 表示目的地代码。 |
| Fields6 | 表示购票张数。 |
| Fields7 | 票种，0 全票，1 军票，2 童票。军票在自动售票机上一般不售 |
| Fields8 | 自动售票机售票员所属客运站代码，如001 |
| Fields9 | 自动售票机售票员工号。 |
| Fields10 | 需要携带的免票儿童数量 如：1，注：一张票只能携带一个免票儿童20160301新增 |
| Fields11 | 实名身份证电话信息串集 第一列为姓名，第二列为电话，第三列为身份证号，第四列为证件类型 格式为：伙子肖1,13891895536,身份证号1,身份证;姓名2,18615673311,身份证号2,居住证;姓名3,18615673311,身份证号3,护照 20160628新增 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** | |
| --- | --- | --- |
| **Object[0]** (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） | |
| Fields2 | 返回信息描述 | |
| **Object[1]** | | |
|  | Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 订单号 |
| Fields2 | 上客运站 |
| Fields3 | 始发站 |
| Fields4 | 班次 |
| Fields5 | 上车日期，如20120220 |
| Fields6 | 目的地 |
| Fields7 | 上车时间，如1330 |
| Fields8 | 座位号 |
| Fields9 | 票价，服务器实际返回的价格 |
| Fields10 | 燃油费，服务器实际返回的价格 |
| Fields11 | 站务费，服务器实际返回的价格 |
| Fields12 | 合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields13 | 1免票儿童0无 20160301新增 |
| Fields14 | 线路名 20160301新增 |
| Fields15 | 乘车人姓名 20160628新增 |
| Fields16 | 乘车人身份证号 20160628新增 |
| Fields17 | 乘车人电话 20160628新增 |
| Fields18 | 证件类型名称 20160628新增 |
| Object[1] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 订单号 |
| Fields2 | 上客运站 |
| Fields3 | 始发站 |
| Fields4 | 班次 |
| Fields5 | 上车日期，如20120220 |
| Fields6 | 目的地 |
| Fields7 | 上车时间，如1330 |
| Fields8 | 座位号 |
| Fields9 | 票价，服务器实际返回的价格 |
| Fields10 | 燃油费，服务器实际返回的价格 |
| Fields11 | 站务费，服务器实际返回的价格 |
| Fields12 | 合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields13 | 1免票儿童0无 |
| Fields14 | 线路名 |
| Fields15 | 乘车人姓名 20160628新增 |
| Fields16 | 乘车人身份证号 20160628新增 |
| Fields17 | 乘车人电话 20160628新增 |
| Fields18 | 证件类型名称 20160628新增 |
| …… | |
| Object[n] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| …… | |

1. 调用存储过程

名 称: AutoSale\_ReserveAtmTickets\_NEW

参数: @ StartStationID @ClassDate @ClassCode @StationID @ArriveID @TicketsSum

@ TicketType @AutoStationID @AutoSalerCode @SChildNum

## 出票接口

接口方法[List<HashMap<String, Object>> WriteAtmTickets（String fields1,String fields2, String fields3,String fields4,String fields5)]

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 订单号，预扣座返回的不同订单号的字符串相加。 |
| Fields2 | 卡号，特指银联卡或自行发行的商务卡,现金支付时为空。 |
| Fields3 | 自动售票机售票员所属客运站代码，如001。 |
| Fields4 | 自动售票机售票员工号。 |
| Fields5 | 支付类型，0银联卡，1商务卡，2现金,5支付宝，6微信7优惠券 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** | |
| --- | --- | --- |
| **Object[0]** (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） | |
| Fields2 | 返回信息描述 | |
| **Object[1]** | | |
|  | Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 票号 |
| Fields2 | 上客运站 |
| Fields3 | 始发站 |
| Fields4 | 班次 |
| Fields5 | 上车日期，如20120220 |
| Fields6 | 目的地 |
| Fields7 | 上车时间，如1330 |
| Fields8 | 车型 |
| Fields9 | 承运人 |
| Fields10 | 票种，0 全票，1 军票，2 童票 |
| Fields11 | 座位号 |
| Fields12 | 票价，服务器实际返回的价格 |
| Fields13 | 燃油费，服务器实际返回的价格 |
| Fields14 | 站务费，服务器实际返回的价格 |
| Fields15 | 合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields16 | 检票口号 |
| Fields17 | 上限票价 |
| Fields18 | 本张票携带的免票儿童数量，如：1 |
| Fields19 | 车号 |
| Fields20 | 线路名 |
| Fields21 | 乘车人姓名 20160628新增 |
| Fields22 | 乘车人身份证号 20160628新增 |
| Fields23 | 乘车人电话 20160628新增 |
| Fields24 | 证件类型名称 20160628新增 |
| Object[1] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 票号 |
| Fields2 | 上客运站 |
| Fields3 | 始发站 |
| Fields4 | 班次 |
| Fields5 | 上车日期，如20120220 |
| Fields6 | 目的地 |
| Fields7 | 上车时间，如1330 |
| Fields8 | 车型 |
| Fields9 | 承运人 |
| Fields10 | 票种，0 全票，1 军票，2 童票 |
| Fields11 | 座位号 |
| Fields12 | 票价，服务器实际返回的价格 |
| Fields13 | 燃油费，服务器实际返回的价格 |
| Fields14 | 站务费，服务器实际返回的价格 |
| Fields15 | 合计价格，字符串，服务端返回四舍五入后的合计整数 |
| Fields16 | 检票口号 |
| Fields17 | 上限票价 |
| Fields18 | 本张票携带的免票儿童数量，如：1 |
| Fields19 | 车号 |
| Fields20 | 线路名 |
| Fields21 | 乘车人姓名 20160628新增 |
| Fields22 | 乘车人身份证号 20160628新增 |
| Fields23 | 乘车人电话 20160628新增 |
| Fields24 | 证件类型名称 20160628新增 |
| …… | |
| Object[n] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| …… | |

1. 调用存储过程

名称: AutoSale\_WriteAtmTickets\_NEW

参数:@BookNO @CardNO @AutoStationID @AutoSalerCode @PayType

## 网上订单查询

接口方法[List<HashMap<String, Object>> QueryAtmBookTicket（String fields1,String fields2,String fields3)]

1. 说明

只能查询到发车日期在当前系统时间之后、未检票、未出客票的订单。

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 取票号，以身份证号作为取票号 |
| Fields2 | 取票密码，以身份证号作为密码 |
| Fields3 | 查询类型 0只按取票人身份证查1按身份证和订单号查2只按订单号查3电子条码查4按乘车人身份证查 20160628新增 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** | |
| --- | --- | --- |
| **Object[0]** (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） | |
| Fields2 | 返回信息描述 | |
| **Object[1]** | | |
|  | Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 上车日期，如20120220 |
| Fields2 | 上车时间，如1330 |
| Fields3 | 上客运站 |
| Fields4 | 班次 |
| Fields5 | 目的地 |
| Fields6 | 座位号 |
| Fields7 | 票价，服务器实际返回的价格 |
| Fields8 | 燃油费，服务器实际返回的价格 |
| Fields9 | 站务费，服务器实际返回的价格 |
| Fields10 | 合计价格 |
| Fields11 | 订票方式 |
| Fields12 | 支付状态，0 已支付，1未支付 |
| Fields13 | 订单号 |
| Fields14 | 检票口号 |
| Fields15 | ID号 20160628新增 |
| Fields16 | 1免票儿童0无 |
| Fields17 | 线路名 20160628新增 |
| Object[1] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 上车日期，如20120220 |
| Fields2 | 上车时间，如1330 |
| Fields3 | 上客运站 |
| Fields4 | 班次 |
| Fields5 | 目的地 |
| Fields6 | 座位号 |
| Fields7 | 票价，服务器实际返回的价格 |
| Fields8 | 燃油费，服务器实际返回的价格 |
| Fields9 | 站务费，服务器实际返回的价格 |
| Fields10 | 合计价格 |
| Fields11 | 订票方式 |
| Fields12 | 支付状态，0 已支付，1未支付 |
| Fields13 | 订单号 |
| Fields14 | 检票口号 |
| Fields15 | ID号 20160628新增 |
| Fields16 | 1免票儿童0无 |
| Fields17 | 线路名 20160628新增 |
| …… | |
| Object[n] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| …… | |

1. 调用存储过程

名称: AutoSale\_QueryAtmBookTicket\_NEW

参数:@IDCard @BookNO @ProcessType

## 网上订单出票

接口方法[List<HashMap<String, Object>> reserveAtmPrintTickets（String fields1,String fields2, String fields3,String fields4,String fields5)]

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 订单号，必须为网上订单查询（QueryAtmBookTicket）返回的订单号。 |
| Fields2 | 卡号，银联卡或自发行的商务卡，可以为空 |
| Fields3 | 自动售票机售票员所属客运站代码，如001。 |
| Fields4 | 自动售票机售票员工号。 |
| Fields5 | 支付类型，0银联卡，1商务卡，2 现金 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** | |
| --- | --- | --- |
| **Object[0]** (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） | |
| Fields2 | 返回信息描述 | |
| **Object[1]** | | |
|  | Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 票号 |
| Fields2 | 上客运站 |
| Fields3 | 始发站 |
| Fields4 | 班次 |
| Fields5 | 上车日期，如20120220 |
| Fields6 | 目的地 |
| Fields7 | 上车时间，如1330 |
| Fields8 | 车型 |
| Fields9 | 承运人 |
| Fields10 | 票种，0 全票，1 军票，2 童票 |
| Fields11 | 座位号 |
| Fields12 | 票价，服务器实际返回的价格 |
| Fields13 | 燃油费，服务器实际返回的价格 |
| Fields14 | 站务费，服务器实际返回的价格 |
| Fields15 | 合计价格 |
| Fields16 | 检票口号 |
| Fields17 | 上限票价 |
| Fields18 | 本张票携带的免票儿童数量，如：1 |
| Fields19 | 车号 |
| Fields20 | 线路名 |
| Object[1] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 票号 |
| Fields2 | 上客运站 |
| Fields3 | 始发站 |
| Fields4 | 班次 |
| Fields5 | 上车日期，如20120220 |
| Fields6 | 目的地 |
| Fields7 | 上车时间，如1330 |
| Fields8 | 车型 |
| Fields9 | 承运人 |
| Fields10 | 票种，0 全票，1 军票，2 童票 |
| Fields11 | 座位号 |
| Fields12 | 票价，服务器实际返回的价格 |
| Fields13 | 燃油费，服务器实际返回的价格 |
| Fields14 | 站务费，服务器实际返回的价格 |
| Fields15 | 合计价格 |
| Fields16 | 检票口号 |
| Fields17 | 上限票价 |
| Fields18 | 本张票携带的免票儿童数量，如：1 |
| Fields19 | 车号 |
| Fields20 | 线路名 |
| …… | |
| Object[n] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| …… | |

1. 调用存储过程

名称: AutoSale\_ReserveAtmPrintTickets\_NEW

参数:@BookNO @CardNO @AutoStationID @AutoSalerCode @PayType @ProcessType

## 车票操作

接口方法[List<HashMap<String, Object>> AtmTicket（String fields1,String fields2, String fields3, String fields4)]

1. 说明

工作人员在自动售票机补票时，将车票初始条码和结束条码上传至票务系统，或者在处理车票卡纸时调整正确的车票条码，并输入票务系统。

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 初始票号 |
| Fields2 | 结束票号 |
| Fields3 | 自动售票机售票员所属客运站代码，如001。 |
| Fields4 | 自动售票机售票员工号。 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** |
| --- | --- |
| Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） |
| Fields2 | 返回信息描述 |

1. 调用存储过程

名称: AutoSale\_AtmTicket

参数:@StartNO @EndNO @AutoStationCode @ AutoSalerCode

## 设备状态

接口方法[List<HashMap<String, Object>> AtmStatus(String fields1,String fields2, String fields3,String fields4,String fields5,String fields6,String fields7,String fields8,String fields9,String field10,String field11)]

1. 说明

自动售票机将自身状态上传至服务器端。

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 自动售票机服务状态，0 正常服务，1 暂停服务 |
| Fields2 | 当前票号 |
| Fields3 | 当前车票纸张总数 |
| Fields4 | 纸币找零模块：1号箱剩余纸币张数 |
| Fields5 | 纸币找零模块：2号箱剩余纸币张数 |
| Fields6 | 纸币找零模块：3号箱剩余纸币张数 |
| Fields7 | 纸币接收模块：纸币钱箱纸币张数 |
| Fields8 | 硬币找零模块：硬币找零斗剩余硬币个数 |
| Fields9 | 状态明细 |
| Fields10 | 自动售票机售票员所属车站代码，如001。 |
| Fields11 | 自动售票机售票员工号。 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** |
| --- | --- |
| Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） |
| Fields2 | 返回信息描述 |

1. 调用存储过程

名称: AutoSale\_AtmStatus

参数:@AutoServiceStatus @CurrentNO @CurrentTicketSum @PaperMoneySum1 @PaperMoneySum2 @PaperMoneySum3 @PaperMoneyReceiveSum @CoinSum @StatusDetial @AutoStationID @AutoSalerCode

## 保单上传

接口方法[List<HashMap<String, Object>> AtmInsurance(String fields1,String fields2, String fields3,String fields4,String fields5,String fields6,String fields7,String fields8,String fields9,String field10,String field11)]

1. 说明

自动售票机将保单信息上传至服务器端。

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 保单号 |
| Fields2 | 被保人姓名 |
| Fields3 | 被保人证件类型 |
| Fields4 | 被保人证件号码 |
| Fields5 | 保费 |
| Fields6 | 保额 |
| Fields7 | 车次 |
| Fields8 | 车票号码 |
| Fields9 | 座位号 |
| Fields10 | 自动售票机售票员所属客运站代码，如001。 |
| Fields11 | 自动售票机售票员工号。 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** |
| --- | --- |
| Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） |
| Fields2 | 返回信息描述 |

1. 调用存储过程

名称: AutoSale\_AtmInsurance

参数:@InsuranceNO @InsurancedName @IDType @IDCard @InsuranceFee @ InsuranceMoney @ClassCode @TicketNO @Seat @AutoStationID @AutoSalerCode

## 查询当前票号

接口方法[List<HashMap<String, Object>> AutoSale\_GetSaleInfo (String fields1,String fields2, String fields3)]

1. 说明

查询某台自动售票机的当前票号。

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 自动售票机售票员所属车站代码，如1004。 |
| Fields2 | 自动售票机售票员工号。 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** |
| --- | --- |
| Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 自动售票机售票员工号 |
| Fields2 | 自动售票机当前票号 |
| Fields2 | 自动售票机当前票张数 |

1. 调用存储过程

名称: AutoSale\_GetSaleInfo\_NEW

参数:@ iOpStation @iOperID

## 查询客运站售票站点

接口方法[List<HashMap<String, Object>> AutoSale\_GetCanSaleArrive(String fields1,String fields2, String fields3)]

1. 输入

| **字段名** | **说明** |
| --- | --- |
| Fields1 | 操作站ID |
| Fields2 | 操作账号ID |
| Fields3 | 到达站简拼 |

1. 输出：以下数据均为字符串对象数组

| **字段名** | **说明** | |
| --- | --- | --- |
| **Object[0]** (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | | |
| Fields1 | 成功标志，0 成功，非0 其他（由票务系统定义返回） | |
| Fields2 | 返回信息描述 | |
| **Object[1]** | | |
|  | Object[0] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 站点ID |
| Fields2 | 站点名称 |
| Fields3 | 简码 |
| Object[1] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| Fields1 | 站点ID |
| Fields2 | 站点名称 |
| Fields3 | 简码 |
| …… | |
| Object[n] (为XML节点即System.Xml.XmlNode[]，以下Fields为该节点元素即XmlElement) | |
| …… | |

1. 调用存储过程

名称: AutoSale\_GetCanSaleArrive\_NEW

参数:@ iOpStation @iOperID @iArrive

# 数据存储设计

每个自动售票机都具有一个唯一编号（DeviceNo），与TP共同维护流水号（DeviceAuditNo）——每次自动售票机签到时，TP返回该自动售票机的最大流水号，所有本地的记录均使用DeviceNo,DeviceAuditNo来唯一确定。

## 购票记录表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 字段说明 | 字段类型 |
| 1 | DeviceID | 设备编号 | Integer |
| 2 | DeviceAuditNo | 设备流水号 | String |
| 3 | TkbTransNo | 票务交易号 | String |
| 4 | DepartCity | 出发城市 | String |
| 5 | DepartStationID | 出发站点 | Integer |
| 7 | DepartStationName | 出发站名称 | String |
| 8 | ArrivalCity | 目的城市 | String |
| 9 | ArrivalStationID | 目的站点 | Integer |
| 10 | ArrivalStationName | 目的站名称 | String |
| 11 | DepartDate | 班次日期 | String |
| 12 | DepartTime | 班车时间 | String |
| 13 | SeatType | 座位类型 | Integer |
| 14 | SeatNo | 座位号 | String |
| 15 | OperationMode | 运营方式 | Integer |
| 16 | ShiftNo | 班次号 | String |
| 17 | TradeType | 业务类型 | Integer |
| 18 | RequestStation | 请求站 | String |
| 19 | BuyingAddress | 操作地址 | String |
| 20 | TicketType | 车票类型：全票，半票，学生票 | Integer |
| 21 | FreeChildrenSeat | 免票儿童座 | Integer |
| 22 | TicketPrice | 车票价格 | BigDecimal |
| 23 | FuelSurcharge | 燃油费 | BigDecimal |
| 24 | StationServiceFee | 站务费 | BigDecimal |
| 25 | TicketNo | 票号 | String |
| 26 | TransResult | 购票结果：0正常，1未出票，2已出未打印 | Integer |

## 投保记录表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 字段说明 | 字段类型 |
| 1 | DeviceID | 设备编号 | Integer |
| 2 | DeviceAuditNo | 设备流水号 | String |
| 3 | TkbTransNo | 票务交易号 | String |
| 7 | SeatNo | 锁票座位号 | String |
| 8 | InsurantNo | 保单号 | String |
| 9 | InsurantPrice | 保费 | BigDecimal |
| 10 | InsurantName | 被保险人的姓名 | String |
| 11 | InsurantIDType | 被保险人的证件类型，身份证/军官证/护照等 | Integer |
| 12 | InsurantIDNo | 被保险人的证件号 | String |
| 13 | InsurantGender | 被保险人的性别 | Integer |
| 14 | InsurantBirthDate | 被保险人的出生日期 | String |
| 15 | AssuredsName | 保单打印受益人的姓名 | String |
| 16 | AssuredsIDType | 受益人的证件类型，身份证/军官证/护照等 | Integer |
| 17 | AssuredsIDNo | 保单打印受益人的证件号 | String |
| 18 | TransTime | 本地购买日期时间 | Datetime |
| 19 | CenterTime | 系统返回日期时间 | Datetime |
| 20 | TransResult | 投保结果：0正常，1未出票，2已出未打印 | Integer |

## 支付日志表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 字段说明 | 字段类型 |
| 1 | DeviceID | 设备编号 | Integer |
| 2 | DeviceAuditNo | 设备流水号 | String |
| 3 | TkbTransNo | 票务交易号 | String |
| 4 | HPayTransNo | 支付交易号 | String |
| 5 | PayType | 支付类型 | String |
| 7 | PayAccountNo | 支付账号 | String |
| 8 | IssueAmount | 票面金额 | BigDecimal |
| 9 | PayAmount | 应付金额 | BigDecimal |
| 10 | TransAmount | 交易金额 | BigDecimal |
| 11 | PaidAmount | 实收金额 | BigDecimal |
| 12 | DiscountAmount | 折扣金额 | BigDecimal |
| 13 | ServiceFee | 手续费 | BigDecimal |
| 14 | BonusAmount | 积分付款的金额 | BigDecimal |
| 15 | PayTime | 支付时间 | Datetime |
| 16 | TransResult | 支付结果：0正常，无其他值 | Integer |
| 17 | TransMessage | 空 | String |

## 异常交易表（凭条）

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 字段名 | 字段说明 | 字段类型 |
| 1 | DeviceID | 设备编号 | Integer |
| 2 | DeviceAuditNo | 设备流水号 | String |
| 3 | TkbTransNo | 票务交易号 | String |
| 4 | HPayTransNo | 支付交易号 | String |
| 5 | PayType | 支付类型 | String |
| 7 | PayAccountNo | 支付账号 | String |
| 8 | IssueAmount | 票面金额 | BigDecimal |
| 9 | PayAmount | 应付金额 | BigDecimal |
| 10 | TransAmount | 交易金额 | BigDecimal |
| 11 | PaidAmount | 实收金额 | BigDecimal |
| 12 | DiscountAmount | 折扣金额 | BigDecimal |
| 13 | ServiceFee | 手续费 | BigDecimal |
| 14 | BonusAmount | 积分付款的金额 | BigDecimal |
| 15 | PayTime | 支付时间 | Datetime |
| 16 | TransResult | 支付结果：0正常，1退钱（未出票），2退票（已出票，人工窗口换票），4退钱（未出保单），8退保（已出保单，人工窗口换票） | String |
| 17 | TransMessage | 凭条内容 |  |