# ACloudAge- CIC软件产品

## 需求分析

根据软件产品实现的功能，反推领域共性需求，可以逐条列举说明。

该部分，作为以产品为核心的设计方案与具体项目间的结合部分，一般要针对具体项目做适应性修改。

因此，该部分可以在某个软件产品方案文档成文后，再进行讨论、抽象，完成该节内容的编写。

## 功能定义

用于描述该软件产品提供的主要功能，可逐条列举说明。

一般而言，功能定义与需求分析中所列出的需求，具有对应关系。也就是说，软件产品所提供的若干个功能点能够满足需求分析小节中提出的领域需求。

## 系统架构

图 ‑1 应用网关系统架构

泛在网络信息接入与集成应用网关的设计采用微内核设计思想，处于运行态的各个子系统模块可以根据用户需要动态插拔，达到良好的扩展性和灵活性。

泛在网络信息接入与集成应用网关系统中主要包括六大核心子系统：1）内核管理器，它作为整个系统的枢纽，负责数据源运行时实例、发布器运行时实例、数据点观察者实例及事件的运行及基本操作处理；2）数据源监控器，它作为系统的核心功能，负责根据具体协议实例与外部数据源建立通信连接，采集外部数据源状态信息及控制数据点，并负责把采集的状态信息服务于内核管理器及数据分析服务子系统；3）数据发布器，负责把数据源监控器采集的数据发布给外部系统或发布rest接口供外部系统访问获得数据；4）事件管理器，定义了事件的类型（系统事件、数据源事件、发布器事件、定时事件及维护事件等），事件的触发及事件的处理（包括：邮件处理方式、设置点处理方式及执行指令处理方式）；5）配置工具，为设计时管理工具，负责配置模型实例，包括：数据源模型配置、数据发布器模型配置、事件模型配置、报表模型配置及系统设置等配置工作；6）数据分析服务，负责数据的报表生成、趋势分析以及历史数据显示的功能。

应用网关系统使用层次化设计，主要包括五层：应用网关接口及常量定义、应用网关设计时模型、应用网关数据库层、应用网关运行时层及应用网关开发服务接口层，如下图所示。



图 ‑2 应用网关系统层次结构

表 ‑1 应用网关系统模块说明

|  | **模块名称** | **模块功能** |
| --- | --- | --- |
|  | 系统接口和常量模块 | 定义应用网关系统的基础接口及常量 |
|  | 数据源数据点设计时模型模块 | 定义应用网关系统的数据源及数据点模块设计时对象模型 |
|  | 事件设计时模型模块 | 定义应用网关系统的事件模块设计时对象模型 |
|  | 发布器设计时模型模块 | 定义应用网关系统的发布器模块设计时对象模型 |
|  | 邮件设计时模型模块 | 定义应用网关系统的邮件模块设计时对象模型 |
|  | 导入导出设计时监控模型模块 | 定义应用网关系统的导入导出模块设计时对象模型 |
|  | 用户设计时模型模块 | 定义应用网关系统的用户模块设计时对象模型 |
|  | 数据库连接初始化模块 | 建立数据库连接以及数据库表的创建 |
|  | 数据库DAO接口模块 | 提供应用网关系统各个模块的DAO操作接口 |
|  | 数据源DAO操作模块 | 提供应用网关系统数据源模块的DAO服务 |
|  | 数据点DAO操作模块 | 提供应用网关系统数据点模块的DAO服务 |
|  | 发布器DAO操作模块 | 提供应用网关系统发布器模块的DAO服务 |
|  | 邮件列表DAO操作模块 | 提供应用网关系统邮件列表模块的DAO服务 |
|  | 系统设置DAO操作模块 | 提供应用网关系统的系统设置模块DAO服务 |
|  | 数据点值DAO操作模块 | 提供应用网关系统数据点值模块的DAO服务 |
|  | 运行时接口模块 | 提供应用网关系统运行时各个模块之间的调用接口 |
|  | 系统运行时管理模块 | 提供应用网关系统在运行时对各个模块的管理服务 |
|  | 数据源运行时管理模块 | 提供应用网关系统数据源模块的运行时管理服务 |
|  | 数据点运行时管理模块 | 提供应用网关系统数据点模块的运行时管理服务 |
|  | 数据点事件运行时管理模块 | 提供应用网关系统数据点事件模块的运行时管理服务 |
|  | 系统事件运行时管理模块 | 提供应用网关系统的系统事件模块运行时管理服务 |
|  | 邮件运行时管理模块 | 提供应用网关系统邮件模块的运行时管理服务 |
|  | 发布器运行时管理模块 | 提供应用网关系统发布器模块的运行时管理服务 |
|  | 导入导出运行时管理模块 | 提供应用网关系统导入导出模块的运行时管理服务 |
|  | view运行时管理模块 | 提供应用网关系统view模块的运行时管理模块 |
|  | 前端服务接口模块 | 提供应用网关系统REST访问服务接口 |
|  | 系统工具包模块 | 提供应用网关系统的工具类，主要包括：异常、同步、线程、校验等工具包 |

应用网关服务系统子在系统设计中，对系统时态进行划分，分为配置时和运行时，如下图所示。配置时指用户的组态过程，通过配置描述模型对象的静态信息及关系，提供配置信息的增加、删除、修改、查找等管理功能，并把配置数据持久化。运行时指根据设计时的配置信息将配置对象实例动态载入内存，启动实例对应服务，如采集、发布、监听等服务。



图 ‑3 应用网关系统应用模式

## 产品组成



图 ‑4 应用网关系统组成

应用网关产品主要组成子系统包括：内核管理器、数据源监控器、数据发布器、事件管理器、数据分析服务及配置工具。

1. 内核管理器

内核管理器负责各个子系统运行控制，作为应用网关枢纽，其主要包括四个核心流程处理：1）模型配置流程，接收用户配置请求并执行持久化；2）数据源实例处理流程，内核管理器主要负责数据源的启动/关闭，处理用户查询/控制请求，并收集状态数据；3）数据发布流程，由内核管理器通过对数据点监听的管理，监听数据推送给发布器，由数据发布器推送给外部系统；4）发布事件流程，应用网关各个子系统提交发布事件请求，由内核管理器发布事件，并调用事件处理方法进行事件处理。

1. 配置工具

应用网关系统采用设计时运行时相分离设计思想，配置工具主要负责系统设计时时态，负责模型实例配置，为运行时系统提供模型实例。

1. 数据源监控器

数据源监控器负责应用网关监控业务，数据源监控建立了业务系统和数据源之间建立基于通信网络的智能通道，提供终端接入、终端信息采集、应用接入、能力开放、通信代理等核心功能，数据源监控器主要是完成不同协议的终端采集，如标准工业协议、通信协议以及自定义协议，及完成不同格式数据的接收处理，如文字、图像、视频等格式数据。数据源监控器完成为数据分析服务及数据发布器提供数据支持。数据源监控器提供了统一数据源扩展接口及数据源快速配置向导。

1. 数据发布器

数据发布器建立了与外部系统（如：GIS服务系统、北斗位置服务系统等）的数据通道，提供了与外部系统更便捷的数据交互。它主要包含两种数据交互方式：推/拉；推主要是数据发布器作为客户端主动与外部系统建立连接，并负责把监听到的数据点信息数据主动推送给外部系统；拉主要是数据发布器作为服务端，监听是否有客户端，当有客户端建立连接时，把数据通过广播的方式发给客户端。数据源发布器同时提供了发布器扩展接口服务。

1. 事件管理器

事件管理器负责应用网关的事件服务，主要事件类型有系统事件、数据源事件、发布器事件、定时事件及维护事件等。它负责事件的产生发布及事件处理，如数据源监控器需要产生一连接异常事件，通过调用事件管理器产生数据源事件，并根据配置的事件处理器进行处理。同时事件管理器还提供了事件类型扩展接口及事件处理器扩展接口。

1. 数据分析服务

数据分析服务主要是对应用网关系统采集的状态数据进行实时分析及显示，包含的服务有报表服务、趋势服务及历史服务。报表服务可以在系统运行时显示在画面上，报表运行后可以打印，或者保存在硬盘上为以后提供打印或显示。也可以使用文本编辑器或者word处理器显示、编辑、打印这些报表。趋势服务无缝的集成了实时和历史数据，趋势服务能够处理大批量的变量数据记录，而且不会降低系统性能和数据完整性。从预置的趋势模板中选择一个合适的模板，在其用户化界面中可以清晰的浏览数据。历史数据服务提供了数据历史数据访问的统一接口服务，可以根据时间、数据源、数据点等关键字进行访问。

## 接口关系

### 外部接口

表 ‑2 应用网关系统外部接口表

| 序号 | 分类 | 类型 | 服务名 | 说明 | 备注 |
| --- | --- | --- | --- | --- | --- |
|  | 用户接口 | 读取接口 | users/getusers | 获取所有用户的信息 |  |
|  | 读取接口 | users/getuser/{id} | 根据用户ID获取用户信息 |  |
|  | 写入接口 | users/saveuser\_admin | 保存管理员 |  |
|  | 写入接口 | users/saveuser | 保存普通用户 |  |
|  | 写入接口 | users/deleteuser/{id} | 根据用户ID删除用户 |  |
|  | 发送接口 | users/send\_testemail | 给用户发送测试邮件 |  |
|  | 读取接口 | users/getuser\_byname/{username} | 根据用户名查询用户 |  |
|  | 写入接口 | users/recordlogin/{userId} | 记录用户登录 |  |
|  | 写入接口 | users/insert\_usercomment | 插入用户注释信息 |  |
|  | 集群模型配置接口 | 写入接口 | cluster/create | 创建Cluster对象 |  |
|  | 写入接口 | cluster/delete/{id} | 根据ID删除Cluster对象 |  |
|  | 写入接口 | cluster/update | 更新Cluster对象 |  |
|  | 读取接口 | cluster/find/{id} | 根据ID查询Cluster对象 |  |
|  | 读取接口 | cluster/findall | 查询所有的Cluster对象 |  |
|  | 写入接口 | networkaddress/create | 创建NetworkAddress对象 |  |
|  | 写入接口 | networkaddress/delete/{id} | 根据ID删除NetworkAddress对象 |  |
|  | 写入接口 | networkaddress/update | 更新NetworkAddress对象 |  |
|  | 读取接口 | networkaddress/find/{id} | 根据ID查询NetworkAddress对象 |  |
|  | 读取接口 | networkaddress/findall | 查询所有的NetworkAddress对象 |  |
|  | 写入接口 | ioserver/create | 创建IoServer对象 |  |
|  | 写入接口 | ioserver/delete/{id} | 根据ID删除IoServer对象 |  |
|  | 写入接口 | ioserver/update | 更新IoServer对象 |  |
|  | 读取接口 | ioserver/find/{id} | 根据ID查询IoServer对象 |  |
|  | 读取接口 | ioserver/findall | 查询所有的IoServer对象 |  |
|  | 写入接口 | reportserver/create | 创建ReportServer对象 |  |
|  | 写入接口 | reportserver/delete/{id} | 根据ID删除ReportServer对象 |  |
|  | 写入接口 | reportserver/update | 更新ReportServer对象 |  |
|  | 读取接口 | reportserver/find/{id} | 根据ID查询ReportServer对象 |  |
|  | 读取接口 | reportserver/findall | 查询所有的ReportServer对象 |  |
|  | 写入接口 | trendserver/create | 创建TrendServer对象 |  |
|  | 写入接口 | trendserver/delete/{id} | 根据ID删除TrendServer对象 |  |
|  | 写入接口 | trendserver/update | 更新TrendServer对象 |  |
|  | 读取接口 | trendserver/find/{id} | 根据ID查询TrendServer对象 |  |
|  | 读取接口 | trendserver/findall | 查询所有的TrendServer对象 |  |
|  | 写入接口 | alarmserver/create | 创建AlarmServer对象 |  |
|  | 写入接口 | alarmserver/delete/{id} | 根据ID删除AlarmServer对象 |  |
|  | 写入接口 | alarmserver/update | 更新AlarmServer对象 |  |
|  | 读取接口 | alarmserver/find/{id} | 根据ID查询AlarmServer对象 |  |
|  | 读取接口 | alarmserver/findall | 查询所有的AlarmServer对象 |  |
|  | 写入接口 | board/create | 创建Board对象 |  |
|  | 写入接口 | board/delete/{id} | 根据ID删除Board对象 |  |
|  | 写入接口 | board/update | 更新Board对象 |  |
|  | 读取接口 | board/find/{id} | 根据ID查询Board对象 |  |
|  | 读取接口 | board/findall | 查询所有的Board对象 |  |
|  | 写入接口 | port/create | 创建Port对象 |  |
|  | 写入接口 | port/delete/{id} | 根据ID删除Port对象 |  |
|  | 写入接口 | port/update | 更新Port对象 |  |
|  | 读取接口 | port/find/{id} | 根据ID查询Port对象 |  |
|  | 读取接口 | port/findall | 查询所有的Port对象 |  |
|  | 写入接口 | iodevice/create | 创建IoDevice对象 |  |
|  | 写入接口 | iodevice/update | 更新IoDevice对象 |  |
|  | 读取接口 | iodevice/find/{id} | 根据ID查询IoDevice对象 |  |
|  | 读取接口 | iodevice/findbyprotocol/{protocol} | 根据协议类型查询IoDevice对象 |  |
|  | 读取接口 | iodevice/findbyxid/{xid} | 根据XID查询IoDevice对象 |  |
|  | 读取接口 | iodevice/findids/{protocol} | 根据协议类型获的所有设备ID |  |
|  | 读取接口 | iodevice/findall | 查询所有的IoDevice |  |
|  | 写入接口 | iodevice/delete/{id} | 根据ID删除IoDevice |  |
|  | 写入接口 | iodevice/deletebyprotocol/{protocol} | 根据协议删除IoDevice |  |
|  | 写入接口 | command/create | 创建Command对象 |  |
|  | 写入接口 | command/delete/{id} | 根据ID删除Command对象 |  |
|  | 写入接口 | command/update | 更新Command对象 |  |
|  | 读取接口 | command/find/{id} | 根据ID查询Command对象 |  |
|  | 读取接口 | command/findall | 查询所有的Command对象 |  |
|  | 写入接口 | variabletag/create | 创建VariableTag对象 |  |
|  | 写入接口 | variabletag/delete/{id} | 根据ID删除VariableTag |  |
|  | 写入接口 | variabletag/deletebyiodeviceid/{ioDeviceId} | 根据IoDevice的ID删除所包含的VariableTags |  |
|  | 写入接口 | variabletag/update | 更新VariableTag对象 |  |
|  | 读取接口 | variabletag/find/{id} | 根据ID查询VariableTag |  |
|  | 读取接口 | variabletag/findbydatatypes | 根据数据类型查询VariableTag |  |
|  | 读取接口 | variabletag/findall | 查询所有的VariableTag对象 |  |
|  | 读取接口 | variabletag/findbyiodeviceid/{ioDeviceId} | 根据IoDevice的ID查询所包含的VariableTags |  |
|  | 读取接口 | variabletag/findbyxid/{xid} | 根据XID查询VariableTag |  |
|  | 读取接口 | variabletag/findids/{ioDeviceId} | 根据IoDevice的ID查询其所包含的VariableTags的IDS |  |
|  | 写入接口 | variabletag/textrenderer | 设置VariableTag的文本格式 |  |
|  | 写入接口 | trendtag/create | 创建TrendTag对象 |  |
|  | 写入接口 | trendtag/delete/{id} | 根据ID删除TrendTag对象 |  |
|  | 写入接口 | trendtag/update | 更新TrendTag对象 |  |
|  | 读取接口 | trendtag/find/{id} | 根据ID查询TrendTag对象 |  |
|  | 读取接口 | trendtag/findall | 查询所有的TrendTag对象 |  |
|  | 数据导入导出接口 | 写入接口 | emport/createexportdata | 创建导出数据 |  |
|  | 读取接口 | emport/importdata | 导出数据 |  |
|  | 读取接口 | emport/getimportstatus | 获得ImportTask的状态 |  |
|  | 发送接口 | emport/cancelimport | 取消ImportTask任务 |  |
|  | 读取接口 | emport/getemportdefinitions | 获取所有的EmportDefinitions |  |
|  | 邮件接口 | 读取接口 | mailinglist/getmailinglists\_all | 查询所有的邮件 |  |
|  | 读取接口 | mailinglist/getmailinglist/{id} | 根据ID获取邮件对象 |  |
|  | 读取接口 | mailinglist/getmailinglist\_byxid/{xid} | 根据XID获取邮件对象 |  |
|  | 写入接口 | mailinglist/savemailinglist | 保存邮件对象 |  |
|  | 写入接口 | mailinglist/deletemailingList/{id} | 根据ID删除邮件对象 |  |
|  | 发送接口 | mailinglist/send\_testemail | 发送测试邮件 |  |
|  | 获取接口 | mailinglist/generate\_unique\_xid | 产生唯一XID |  |
|  | 模块管理接口 | 写入接口 | modules/delete/{moduleName} | 根据模块名称删除模块 |  |
|  |  | 写入接口 | modules/schedule\_restart | 生成TimeoutTask实例 |  |
|  | 写入接口 | modules/versioncheck | 版本校验 |  |
|  | 写入接口 | modules/download | 下载版本更新 |  |
|  | 获取接口 | modules/downloads\_isfinished | 获取下载是否完成 |  |
|  | 获取接口 | modules/downloadsresults | 下载结果 |  |
|  | 获取接口 | modules/getmodules | 获取所有的模块 |  |
|  | 获取接口 | modules/getcoremodule | 获取core模块 |  |
|  | 获取接口 | modules/getdefinitions/{defenitionType} | 获取具体类型的所有ModuleElementDefinitions |  |
|  | 获取接口 | modules/getmenuitems/{role} | 根据角色去获取菜单项 |  |
|  | 获取接口 | modules/ismoduleexists/{moduleName} | 根据模块名称判断该模块是否存在 |  |
|  | 获取接口 | modules/getmodule/{moduleName} | 根据模块名称获取模块 |  |
|  | 点连接接口 | 获取接口 | pointlinkcontroller/getsourcevartags | 获取源数据点 |  |
|  | 获取接口 | pointlinkcontroller/gettargetvartags | 获取目标数据点 |  |
|  | 获取接口 | pointlinkcontroller/getpointlinks | 获取所有的点连接 |  |
|  | 获取接口 | pointlinkcontroller/getpointlink/{id} | 根据ID获取点连接 |  |
|  | 写入接口 | pointlinkcontroller/create | 创建点连接 |  |
|  | 写入接口 | pointlinkcontroller/delete/{id} | 根据ID删除点连接 |  |
|  | 写入接口 | pointlinkcontroller/validatescript | 校验script |  |
|  | 发布器接口 | 获取接口 | publisher/getpublishertypes | 获取publisher的所有的类型 |  |
|  | 获取接口 | publisher/getpublishers | 获取所有的publisher |  |
|  | 写入接口 | publisher/togglepublisher/{publisherId} | 启动publisher |  |
|  | 写入接口 | publisher/deletepublisher/{publisherId} | 根据ID删除publisher |  |
|  | 写入接口 | publisher/canceltest | 取消测试 |  |
|  | 获取接口 | publisher/getpublishervo | 获取publisher设计时对象 |  |
|  | 获取接口 | publisher/getpendingevents | 获取publisher的事件实例 |  |
|  | Sql Console接口 | 读取接口 | sqlconsole/executesqlquery | 执行SQL查询操作 |  |
|  | 写入接口 | sqlconsole/executesqlupdate | 执行SQL更新操作 |  |
|  | 读取接口 | sqlconsole/gettablelist | 获取表列表 |  |
|  | 系统配置接口 | 读取接口 | systemsettings/getinstancedescsetting | 获得实例描述设置 |  |
|  | 读取接口 | systemsettings/getemailsettings | 获得邮件相关设置 |  |
|  | 读取接口 | systemsettings/geteventtypesettings | 获得事件类型设置 |  |
|  | 读取接口 | systemsettings/gethttpsettings | 获得Http相关设置 |  |
|  | 读取接口 | systemsettings/geteventpurgesettings | 获得事件清理设置 |  |
|  | 读取接口 | systemsettings/getuiperformancesetting | 获得UI设置 |  |
|  | 读取接口 | systemsettings/getfuturedatelimitperiod | 获得时间周期设置 |  |
|  | 读取接口 | systemsettings/getlanguage | 获得语言设置 |  |
|  | 读取接口 | systemsettings/getcolour | 获得颜色相关设置 |  |
|  | 读取接口 | systemsettings/getdatabasesize | 获得数据库目录大小 |  |
|  | 读取接口 | systemsettings/getfiledatacount | 获得所有文件数量 |  |
|  | 读取接口 | systemsettings/getfiledatasize | 获得所有文件大小 |  |
|  | 读取接口 | systemsettings/getpointhistorycountlist | 获得历史点数量对象的list |  |
|  | 读取接口 | systemsettings/getpointhistorycountsum | 获得历史点数量总数 |  |
|  | 读取接口 | systemsettings/geteventcount | 获得事件数量 |  |
|  | 写入接口 | systemsettings/saveemailsettings | 保存邮件设置 |  |
|  | 写入接口 | systemsettings/sendtestemail | 发送测试邮件 |  |
|  | 写入接口 | systemsettings/savesystemeventalarmlevels | 保存系统事件报警级别 |  |
|  | 写入接口 | systemsettings/saveauditeventalarmlevels | 保存审计事件报警级别 |  |
|  | 写入接口 | systemsettings/savehttpsettings | 保存Http设置 |  |
|  | 写入接口 | systemsettings/savemiscsettings | 保存其他设置 |  |
|  | 写入接口 | systemsettings/saveinfosettings | 保存实例设置 |  |
|  | 写入接口 | systemsettings/savecoloursettings | 保存颜色设置 |  |
|  | 写入接口 | systemsettings/savelanguagesettings | 保存语言设置 |  |
|  | 写入接口 | systemsettings/purgenow | 立即清理数据 |  |
|  | 写入接口 | systemsettings/purgealldata | 清理系统所有历史数据 |  |
|  | 写入接口 | systemsettings/savesettings | 保存所有设置信息 |  |
|  | 读取接口 | systemsettings/getvalue | 根据KEY和默认Value值获取设置Value |  |
|  | 读取接口 | systemsettings/getintvalue/{key} | 根据KEY值获取设置Value |  |
|  | 读取接口 | systemsettings/getintvaluewithdefaultvalue | 根据默认Value获取int Value |  |
|  | 读取接口 | systemsettings/getbooleanvalue/{key} | 根据KEY获取bool类型的Value |  |
|  | 读取接口 | systemsettings/getsystemsettingsdefinitions | 获取所有的SystemSettingsDefinitions |  |
|  | 读取接口 | systemsettings/getdatabasetypename | 获取数据库类型名称 |  |
|  | 数据点详细信息接口 | 读取接口 | variabletagdetails/getvartagbyxid/{xid} | 通过XID获取VariableTag |  |
|  | 读取接口 | variabletagdetails/getuservariabletags | 获取当前用户权限下的VariableTags |  |
|  | 读取接口 | variabletagdetails/getvartagaccesstype | 获取当前用户对VariableTag的操作权限 |  |
|  | 读取接口 | variabletagdetails/getlatestpointvaluetimes | 获取点的最新值 |  |
|  | 读取接口 | variabletagdetails/getvartagwithsetperm/{id} | 通过ID获取VariableTag，并且用户对其需有set权限 |  |
|  | 读取接口 | variabletagdetails/getvartagwithreadperm/{id} | 通过ID获取VariableTag，并且用户对其需有read权限 |  |
|  | 读取接口 | variabletagdetails/getvartagbyxidwithsetperm/{xid} | 通过XID获取VariableTag，并且用户对其需有set权限 |  |
|  | 读取接口 | variabletagdetails/getvartagbyxidwithreadperm/{xid} | 通过XID获取VariableTag，并且用户对其需有read权限 |  |
|  | 读取接口 | variabletagdetails/getdatapointrt/{id} | 通过ID获取DataPointRT |  |
|  | 写入接口 | variabletagdetails/forcepointread | 通过ID设置VariableTag可读 |  |
|  | 读取接口 | variabletagdetails/generatevartagxid | 产生VariableTag的XID |  |
|  | 读取接口 | variabletagdetails/generateiodevicexid | 产生IoDevice的XID |  |
|  | 事件处理接口 | 读取接口 | eventhandler/geteventtypes | 根据协议类型获得该协议设备的事件类型 |  |
|  | 读取接口 | eventhandler/geteventhandlers | 根据EventTypeVo获取它的处理方式 |  |
|  | 读取接口 | eventhandler/getusereventtypelist | 获取所有UserEventTypeVO实例 |  |
|  | 读取接口 | eventhandler/getsystemevents | 获取所有SystemEventTypeVO实例 |  |
|  | 读取接口 | eventhandler/getauditevents | 获取所有AuditEventTypeVO实例 |  |
|  | 读取接口 | eventhandler/getadmineventtypes | 获取所有AdminEventTypeVO实例 |  |
|  | 写入接口 | eventhandler/savesetpointeventhandler | 保存点事件处理实例 |  |
|  | 写入接口 | eventhandler/saveemaileventhandler | 保存邮件事件处理实例 |  |
|  | 写入接口 | eventhandler/saveprocesseventhandler | 保存处理事件处理实例 |  |
|  | 写入接口 | eventhandler/delete/{id} | 根据ID删除事件处理实例 |  |
|  | 写入接口 | eventhandler/testprocesscommand | 测试命令执行 |  |
|  | 读取接口 | eventhandler/getdatavalue | 获取数据点的Value |  |
|  | 读取接口 | eventhandler/geteventtypedefinitions | 获取EventTypeDefinitions |  |
|  | Graphical Views接口 | 读取接口 | graphicalviews/imagesets | 获取所有注册的图片集 |  |
|  | 读取接口 | graphicalviews/imageset/{id} | 获取指定ID的图片集 |  |
|  | 读取接口 | graphicalviews/dynamicimages | 获取所有注册的动态图片 |  |
|  | 读取接口 | graphicalviews/dynamicimage/{id} | 获取指定ID的动态图片 |  |
|  | 读取接口 | graphicalviews/view/{id} | 获取指定ID的GraphicalView，并验证用户是否具有相关权限 |  |
|  | 读取接口 | graphicalviews/viewbyxid/{xid} | 获取指定xid的GraphicalView，并验证用户是否具有相关权限 |  |
|  | 读取接口 | graphicalviews/viewbyname/{name} | 获取指定name的GraphicalView，并验证用户是否具有相关权限 |  |
|  | 写入接口 | graphicalviews/saveview | 保存View |  |
|  | 写入接口 | graphicalviews/delete/{id} | 删除View |  |
|  | 读取接口 | graphicalviews/viewnames | 获取当前登录user可操作的view名称 |  |
|  | 读取接口 | graphicalviews/uniquegraphicalviewxid | 获取一个唯一的GraphicalView的xid |  |
|  | 读取接口 | graphicalviews/componenttypes | 获取所有的组件类型 |  |
|  | 写入接口 | graphicalviews/deleteviewshare | 删除用户对于指定view的权限 |  |
|  | 写入接口 | graphicalviews/validate\_variabletag | 校验用户是否有对VariableTag读的权限 |  |
|  | 事件维护接口 | 读取接口 | maintenanceevent/findall | 获取所有的事件维护实例 |  |
|  | 读取接口 | maintenanceevent/find/{id} | 通过ID查询事件维护实例 |  |
|  | 读取接口 | maintenanceevent/iseventactive/{id} | 事件维护实例是否active |  |
|  | 写入接口 | maintenanceevent/create | 创建事件维护实例 |  |
|  | 写入接口 | maintenanceevent/delete/{id} | 根据ID删除事件维护实例 |  |
|  | 写入接口 | maintenanceevent/togglemaintenanceevent/{id} | 启动一个事件维护实例 |  |
|  | 读取接口 | maintenanceevent/generateuniquexid | 获取一个唯一的XID |  |
|  | 定时事件接口 | 读取接口 | scheduledevent/findall | 获取所有的定时事件 |  |
|  | 读取接口 | scheduledevent/find/{id} | 通过ID获取定时事件 |  |
|  | 写入接口 | scheduledevent/create | 创建定时事件 |  |
|  | 写入接口 | scheduledevent/delete/{id} | 根据ID删除定时事件 |  |
|  | 报表接口 | 读取接口 | report/getreports/{userId} | 根据USERID获取该用户下的ReportVO |  |
|  | 读取接口 | report/getreport | 根据ID获取ReportVO |  |
|  | 写入接口 | report/savereport | 添加ReportVO |  |
|  | 写入接口 | report/runreport | 运行ReportVO对象 |  |
|  | 写入接口 | report/deletereport/{id} | 根据ID删除ReportVO对象 |  |
|  | 写入接口 | report/deletereportinstance{instanceId} | 根据实例ID删除ReportVO的运行实例 |  |
|  | 读取接口 | report/getreportinstances{userId} | 根据用户ID删除ReportVO的运行实例 |  |
|  | 写入接口 | report/setpreventpurge | ReportVO运行实例设置是否清除数据参数 |  |
|  | 写入接口 | report/createreport\_fromwatchlist | 给WatchList的点创建报表 |  |
|  | 读取接口 | report/getreportinstance | 获取报表运行实例 |  |
|  | Watchlist接口 | 读取接口 | watchlist/get\_watchlist\_pointhierarchy | 获取WatchList的点继承结构 |  |
|  | 写入接口 | watchlist/preparewatchlist/{watchListId} | WatchList包含的Points属性设置为可设置的 |  |
|  | 读取接口 | watchlist/getselectedwatchlist/{userId} | 获得用户选择的WatchList |  |
|  | 读取接口 | watchlist/getwatchlists/{userId} | 获取用户所拥有的WatchLists |  |
|  | 写入接口 | watchlist/createnewwatchlist | 创建一个WatchList |  |
|  | 写入接口 | watchlist/savewatchlist | 存储一个WatchList |  |
|  | 写入接口 | watchlist/saveselectedwatchlist | 保存用户选择的WatchList |  |
|  | 写入接口 | watchlist/updatewatchlistname | 更新WatchList的name |  |
|  | 写入接口 | watchlist/addnewwatchlist | 添加一个新的WatchList |  |
|  | 写入接口 | watchlist/deletewatchlist/{watchListId} | 根据ID删除WatchList |  |
|  | 写入接口 | watchlist/setselectedwatchlist/{watchListId} | 设置选择WatchList中所有点为可设置的 |  |
|  | 读取接口 | watchlist/getwatchlistpointids/{watchListId} | 获取WatchList所有点的ID |  |
|  | 读取接口 | watchlist/getwatchlistpoints/{watchListId} | 获取WatchList的所有点 |  |
|  | 写入接口 | watchlist/addtowatchlist | 添加点到WatchList中 |  |
|  | 写入接口 | watchlist/addupdateshareduser | WatchList中添加SharedUser |  |
|  | 写入接口 | watchlist/removeshareduser | 删除WatchList中的SharedUser |  |
|  | 写入接口 | watchlist/removefromwatchlist | 从WatchList中删除VariableTag |  |
|  | 写入接口 | watchlist/moveup | WatchList中指定VariableTag上移 |  |
|  | 写入接口 | watchlist/movedown | WatchList中指定VariableTag下移 |  |
|  | 读取接口 | watchlist/getwatchlistbyid/{watchListId} | 根据WatchList的ID获得WatchList |  |
|  | 本地变量处理接口 | 写入接口 | localvariabletag/create | 创建本地变量 |  |
|  | 写入接口 | localvariabletag/delete/{id} | 根据ID删除本地变量 |  |
|  | 写入接口 | localvariabletag/update | 更新本地变量 |  |
|  | 读取接口 | localvariabletag/find/{id} | 根据ID获取本地变量 |  |
|  | 读取接口 | localvariabletag/findall | 获取所有的本地变量 |  |
|  | 读取接口 | localvariabletag/generatexid | 生成唯一XID |  |

### 内部接口

表 ‑3 应用网关系统内部接口表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 配置工具 | 内核管理器 | saveIoDevice | 保存IoDevice | RuntimeManager.saveIoDevice() | IoDevice | 需要存储的IoDevice对象 |
|  | 配置工具 | 内核管理器 | savePublisher | 保存Publisher | RuntimeManager.savePublisher() | Publisher | 需要存储的Publisher对象 |
|  | 配置工具 | 内核管理器 | saveVariableTag | 保存数据点 | RuntimeManager.saveVariableTag() | VariableTag | 需要存储的VariableTag对象 |
|  | 配置工具 | 内核管理器 | updateIoDevice | 更新IoDevice | RuntimeManager.updateIoDevice() | IoDevice | 需要更新的IoDevice对象 |
|  | 配置工具 | 内核管理器 | updateVariableTag | 更新数据点 | RuntimeManager.updateVariableTag() | VariableTag | 需要更新的VariableTag对象 |
|  | 配置工具 | 内核管理器 | getDataPoint | 查询数据点 | RuntimeManager.getDataPoint() | String | 查询数据点的ID |
|  | 配置工具 | 内核管理器 | getPublisher | 查询Publisher | RuntimeManager.getPublisher() | String | 查询Publisher的ID |
|  | 配置工具 | 内核管理器 | deleteIoDevice | 删除IoDevice | RuntimeManager.deleteIoDevice() | String | 删除IoDevice的ID |
|  | 配置工具 | 内核管理器 | deletePublisher | 删除Publisher | RuntimeManager.deletePublisher() | String | 删除Publisher的ID |
|  | 配置工具 | 内核管理器 | deleteVariableTag | 删除数据点 | RuntimeManager.deleteVariableTag() | String | 删除数据点的ID |
|  | 数据源监控器 | 内核管理器 | initializeDataSource | 初始化数据源 | RuntimeManager.initializeDataSource() | IoDevice | 需要初始化的IoDevice对象 |
|  | 数据源监控器 | 内核管理器 | startDataPoint | 开始数据点 | RuntimeManager.startDataPoint() | VariableTag | 需要启动的VariableTag对象 |
|  | 数据源监控器 | 内核管理器 | startIoDevicePolling | 开始数据源轮询 | RuntimeManager.startIoDevicePolling() | IoDevice | 需要启动的IoDevice对象 |
|  | 数据源监控器 | 内核管理器 | stopDataPoint | 停止数据点 | RuntimeManager.stopDataPoint() | String | 需要停止数据点的ID |
|  | 数据源监控器 | 内核管理器 | stopDataSource | 停止数据源 | RuntimeManager.stopDataSource() | String | 需要停止数据源的ID |
|  | 数据源监控器 | 内核管理器 | setDataPointValue | 单点控制数据点 | RuntimeManager.setDataPointValue() | String | 控制点的ID |
| DataValue | 数据点控制值 |
| SetPointSource | 控制点的出处 |
|  | 数据源监控器 | 内核管理器 | setBatchPointValues | 多点控制数据点 | RuntimeManager.setBatchPointValues() | Map<String, String> | Map存放的Key/Value键值对，Key表示控制点的ID，Value表示控制值 |
| SetPointSource | 控制点的出处 |
|  | 数据源监控器 | 内核管理器 | addDataPointListener | 添加数据点监听者 | RuntimeManager.addDataPointListener() | String | 数据点的ID |
| DataPointListener | 数据点的监听者 |
|  | 数据源监控器 | 内核管理器 | removeDataPointListener | 删除数据点监听者 | RuntimeManager.removeDataPointListener() | String | 数据点的ID |
| DataPointListener | 数据点的监听者 |
|  | 数据源监控器 | 内核管理器 | getDataPointListeners | 获取数据点监听者 | RuntimeManager.getDataPointListeners() | 无 | 无 |
|  | 数据源监控器 | 内核管理器 | joinTermination | 联合停止 | RuntimeManager.joinTermination() | 无 | 无 |
|  | 数据发布器 | 内核管理器 | startPublisher | 启动数据发布器 | RuntimeManager.startPublisher() | PublisherVO | 需要启动的PublisherVO对象 |
|  | 数据发布器 | 内核管理器 | stopPublisher | 停止数据发布器 | RuntimeManager.stopPublisher() | String | 停止发布器的ID |
|  | 数据发布器 | 内核管理器 | getRunningPublisher | 根据ID获取正在运行的发布器 | RuntimeManager.getRunningPublisher() | String | 发布器的ID |
|  | 数据发布器 | 内核管理器 | getRunningPublishers | 获取正在运行的所有发布器 | RuntimeManager.getRunningPublishers() | 无 | 无 |
|  | 事件管理器 | 内核管理器 | addListener | 添加事件管理器监听者 | RuntimeManager.addListener() | EventManagerListenerDefinition | 事件管理器监听者对象 |
|  | 事件管理器 | 内核管理器 | removeListener | 删除事件管理器监听者 | RuntimeManager.removeListener() | EventManagerListenerDefinition | 事件管理器监听者对象 |
|  | 事件管理器 | 内核管理器 | cancelEventsForDataPoint | 取消指定数据点的事件 | RuntimeManager.cancelEventsForDataPoint () | String | 数据点的ID |
|  | 事件管理器 | 内核管理器 | cancelEventsForDataSource | 取消指定数据源的事件 | RuntimeManager.cancelEventsForDataSource() | String | 数据源的ID |
|  | 事件管理器 | 内核管理器 | cancelEventsForPublisher | 取消指定发布器的事件 | RuntimeManager.cancelEventsForPublisher() | String | Publisher的ID |
|  | 事件管理器 | 内核管理器 | initialize | 初始化活动事件 | RuntimeManager.initialize() | 无 | 无 |
|  | 事件管理器 | 内核管理器 | raiseEvent | 产生事件 | RuntimeManager.raiseEvent() | EventType | 事件类型 |
| long | 事件时间 |
| boolean | 是否可返回普通 |
| int | 事件报警级别 |
| TranslatableMessage | 事件消息描述 |
| Map<String, Object> | 事件上下文 |
|  | 事件管理器 | 内核管理器 | returnToNormal | 事件变为普通 | RuntimeManager.returnToNormal() | EventType | 事件类型 |
| long | 时间 |
|  | 事件管理器 | 内核管理器 | returnToNormal | 事件变为普通 | RuntimeManager.returnToNormal() | EventType | 事件类型 |
| long | 时间 |
| int | 事件变为普通的原因 |
|  | 数据分析服务 | 数据源监控器 | getVariableTag | 根据ID获取VariableTag | VariableTagDetailsController.getVariableTag() | String | VariableTag的ID |
|  | 数据分析服务 | 数据源监控器 | getVariableTagByXid | 根据ID获取VariableTag | VariableTagDetailsController.getVariableTagByXid() | String | VariableTag的XID |
|  | 数据分析服务 | 数据源监控器 | getVariableTagByXidWithReadPermission | 根据XID获取有读权限的VariableTag | VariableTagDetailsController.getVariableTagByXidWithReadPermission() | String | VariableTag的XID |
|  | 数据分析服务 | 内核管理器 | getVariableTagWithReadPermission | 根据ID获取有读权限的VariableTag | VariableTagDetailsController.getVariableTagWithReadPermission() | String | VariableTag的ID |
|  | 数据分析服务 | 内核管理器 | getLatestPointValueTimes | 获取数据点最近的n个数据值 | PointValuesResource.getLatestPointValueTimes() | String | VariableTag的ID |
| int | 最新值的个数 |
|  | 数据分析服务 | 内核管理器 | getPointValueAt | 获取数据点在某一时刻数据点值 | PointValuesResource.getPointValueAt() | String | VariableTag的ID |
| long | 时刻 |
|  | 数据分析服务 | 内核管理器 | getPointValueBefore | 获取数据点在某一时刻之前的数据值 | PointValuesResource.getPointValueBefore() | String | VariableTag的ID |
| long | 时刻 |
|  | 数据分析服务 | 内核管理器 | getPointValues | 获取数据点在某一时刻以来的数据值 | PointValuesResource.getPointValues() | String | VariableTag的ID |
| long | 时刻 |
|  | 数据分析服务 | 内核管理器 | getPointValuesBetween | 获取数据点在某一时间范围之内的数据值 | PointValuesResource.getPointValuesBetween() | String | 需要更新的VariableTag对象 |
| long | 起始时刻 |
| long | 终止时刻 |

## 业务流程

应用网关系统业务流程主要包含设计时配置业务流程和运行时业务流程，如下图所示。



**图 1‑5 应用网关业务流程**

业务网关系统的业务流程说明如下：

1. 设计时
   1. 用户发送模型配置请求，并提交模型配置参数给配置工具；
   2. 配置工具把参数组织为模型对象并把请求提交给内核管理器；
   3. 由内核管理器持久化模型数据；
2. 运行时
   1. 在系统启动中，由内核管理器处于使能状态的模型实例载入内存中，并对实例进行初始化；
   2. 用户发送查询/控制请求给内核管理器；
   3. 内核管理器收到请求以后，首先判断数据源是否为使能，若未使能，先对数据源进行使能，然后把查询/控制请求发送给具体数据源；
   4. 数据源接收请求，若为查询请求，则对外部数据源数据点值进行查询，若为控制请求，则对外部数据源的数据点值进行控制；
   5. 数据源实例负责接收外部数据源的返回响应，对其进行解析入库，若与外部数据源连接或数据出现异常，则发送异常消息请求给内核管理器，由内核管理器调用事件管理器进行事件的发布，并把事件消息返回给用户同时进行入库操作；
   6. 数据源实例同时把数据源状态信息返回给内核管理器；
   7. 内核管理器首先把状态信息数据返回给客户端，若有发布器对数据点进行监听，内核管理器还需把数据变化消息告知发布器；
   8. 数据源监听到有新状态信息时，把新状态信息推送给外部系统，若连接或数据推送发送异常，则由数据发布器发送触发事件请求到内核管理器，内核管理器再调用事件管理器产生事件，把事件消息返回给用户并进行入库操作。

## 研究内容

### 内核管理器

#### 软件架构



**图 1‑6 内核管理器软件架构**

内核管理器软件负责应用网关的核心流程控制，作为系统运行时控制器。在系统启动时，内核管理器负责Module的加载及实例的加载；实例加载成功后，负责进行实例初始化，恢复实例上次运行状态，并发布系统统一rest访问接口。当系统启动后，用户通过统一访问接口提交查询/控制请求到数据源管理器，由数据源管理器发送给数据源管理器，若有异常事件则调用事件触发器进行事件发布；当数据源管理器到状态数据/响应，把数据返回给用户，同时给该数据点的监听者，然后由发布器管理器选择合适的发布器推送给外部系统。

#### 系统组成



**图 1‑7 内核管理器系统组成**

内核管理器系统主要由以下几个模块组成：

1. 数据源管理器

数据源管理器主要负责运行时数据源实例统一管理，负责主要业务包括数据源的启动/停止及数据源的查询/控制。它接收到用户请求后，寻找对应的数据源实例，再把请求交给数据源处理。

1. Module管理器

Module管理器负责除核心模块之外的所有扩展模块的统一管理，它负责模块的加载及卸载，并且提供了扩展接口，如需要添加license校验模块，只需要实现对应扩展接口就可以纳入Module的管理。

1. 监听者管理器

监听者管理主要是对数据点的监听者进行管理，负责添加和移除监听者，其监听数据点的行为通常包含数据点的初始化、数据点值的更新、数据点值的改变、数据点的停止、多数据点的值改变及多数据段的值更新。

1. 发布器管理器

发布器管理器是对运行时发布器实例进行统一管理，负责的主要业务为发布器的启动/关闭及发布器实例的添加和卸载。

1. 事件触发器

事件触发器的主要业务是负责接收系统其他模块的事件请求，并通知事件处理模块产生事件及对产生事件作出相应的处理。

#### 接口关系

##### 外部接口

表 ‑4 内核管理器系统外部接口表

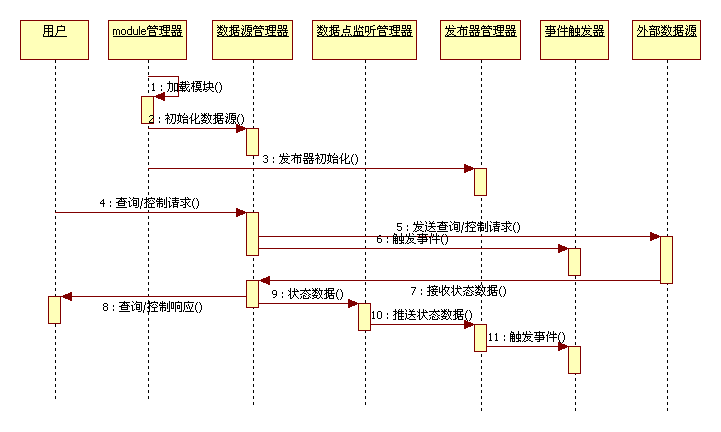
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 客户端 | 内核管理器 | initializeDataSource | 初始化数据源 | RuntimeManager.initializeDataSource() | IoDevice | 需要初始化的IoDevice对象 |
|  | 客户端 | 内核管理器 | startDataPoint | 开始数据点 | RuntimeManager.startDataPoint() | VariableTag | 需要启动的VariableTag对象 |
|  | 客户端 | 内核管理器 | startIoDevicePolling | 开始数据源轮询 | RuntimeManager.startIoDevicePolling() | IoDevice | 需要启动的IoDevice对象 |
|  | 客户端 | 内核管理器 | stopDataPoint | 停止数据点 | RuntimeManager.stopDataPoint() | String | 需要停止数据点的ID |
|  | 客户端 | 内核管理器 | stopDataSource | 停止数据源 | RuntimeManager.stopDataSource() | String | 需要停止数据源的ID |
|  | 客户端 | 内核管理器 | setDataPointValue | 单点控制数据点 | RuntimeManager.setDataPointValue() | String | 控制点的ID |
| DataValue | 数据点控制值 |
| SetPointSource | 控制点的出处 |
|  | 客户端 | 内核管理器 | startPublisher | 启动数据发布器 | RuntimeManager.startPublisher() | PublisherVO | 需要启动的PublisherVO对象 |
|  | 客户端 | 内核管理器 | stopPublisher | 停止数据发布器 | RuntimeManager.stopPublisher() | String | 停止发布器的ID |
|  | 客户端 | 内核管理器 | cancelEventsForDataPoint | 取消指定数据点的事件 | RuntimeManager.cancelEventsForDataPoint () | String | 数据点的ID |
|  | 客户端 | 内核管理器 | cancelEventsForDataSource | 取消指定数据源的事件 | RuntimeManager.cancelEventsForDataSource() | String | 数据源的ID |
|  | 客户端 | 内核管理器 | cancelEventsForPublisher | 取消指定发布器的事件 | RuntimeManager.cancelEventsForPublisher() | String | Publisher的ID |
|  | 客户端 | 内核管理器 | returnToNormal | 事件变为普通 | RuntimeManager.returnToNormal() | EventType | 事件类型 |
| long | 时间 |
|  | 客户端 | 内核管理器 | returnToNormal | 事件变为普通 | RuntimeManager.returnToNormal() | EventType | 事件类型 |
| long | 时间 |
| int | 事件变为普通的原因 |
|  | 内核管理器 | 数据源监控器 | initialize | 启动数据点 | IDataPointRT.initialize() | 无 | 无 |
|  | 内核管理器 | 数据源监控器 | beginPolling | 开始数据源轮询 | IDataSourceRT.beginPolling() | 无 | 无 |
|  | 内核管理器 | 数据源监控器 | terminate | 停止数据点 | IDataPointRT.terminate() | 无 | 无 |
|  | 内核管理器 | 数据源监控器 | terminate | 停止数据源 | IDataSourceRT.terminate() | 无 | 无 |
|  | 内核管理器 | 数据源监控器 | setPointValue | 设置数据点值 | IDataSourceRT.setPointValue() | IDataPointRT | 需要设置值的数据点对象 |
| PointValueTime | 数据点值 |
| SetPointSource | 设置点源头 |
|  | 内核管理器 | 数据源监控器 | setBatchPointValues | 多点控制 | IDataSourceRT.setBatchPointValues() | Map<IDataPointRT, PointValueTime> | 多点组成的Map |
| SetPointSource | 设置点值源头 |
|  | 内核管理器 | 数据发布器 | initialize | 初始化发布器 | IPublisherRT.initialize() | 无 | 无 |
|  | 内核管理器 | 数据发布器 | terminate | 停止发布器 | IPublisherRT.terminate() | 无 | 无 |
|  | 内核管理器 | 事件管理器 | raiseEvent | 产生事件 | SystemEventType.raiseEvent() | EventType | 事件类型 |
| long | 事件时间 |
| boolean | 是否可返回普通 |
| TranslatableMessage | 事件消息描述 |
|  | 内核管理器 | 事件管理器 | returnToNormal | 事件变为普通 | EventInstance.returnToNormal() | long | 时间 |
| int | 事件变为普通的原因 |
|  | 配置工具 | 内核管理器 | saveIoDevice | 保存IoDevice | RuntimeManager.saveIoDevice() | IoDevice | 需要存储的IoDevice对象 |
|  | 配置工具 | 内核管理器 | savePublisher | 保存Publisher | RuntimeManager.savePublisher() | Publisher | 需要存储的Publisher对象 |
|  | 配置工具 | 内核管理器 | saveVariableTag | 保存数据点 | RuntimeManager.saveVariableTag() | VariableTag | 需要存储的VariableTag对象 |
|  | 配置工具 | 内核管理器 | updateIoDevice | 更新IoDevice | RuntimeManager.updateIoDevice() | IoDevice | 需要更新的IoDevice对象 |
|  | 配置工具 | 内核管理器 | updateVariableTag | 更新数据点 | RuntimeManager.updateVariableTag() | VariableTag | 需要更新的VariableTag对象 |
|  | 配置工具 | 内核管理器 | getDataPoint | 查询数据点 | RuntimeManager.getDataPoint() | String | 查询数据点的ID |
|  | 配置工具 | 内核管理器 | getPublisher | 查询Publisher | RuntimeManager.getPublisher() | String | 查询Publisher的ID |
|  | 配置工具 | 内核管理器 | deleteIoDevice | 删除IoDevice | RuntimeManager.deleteIoDevice() | String | 删除IoDevice的ID |
|  | 配置工具 | 内核管理器 | deletePublisher | 删除Publisher | RuntimeManager.deletePublisher() | String | 删除Publisher的ID |
|  | 配置工具 | 内核管理器 | deleteVariableTag | 删除数据点 | RuntimeManager.deleteVariableTag() | String | 删除数据点的ID |
|  | 内核管理器 | 数据源监控器 | saveIoDevice | 保存IoDevice | IIoDeviceDao.saveIoDevice() | IoDevice | 需要存储的IoDevice对象 |
|  | 内核管理器 | 数据发布器 | savePublisher | 保存Publisher | PublisherDao.savePublisher() | Publisher | 需要存储的Publisher对象 |
|  | 内核管理器 | 数据源监控器 | saveVariableTag | 保存数据点 | IVariableTagDao.saveVariableTag() | VariableTag | 需要存储的VariableTag对象 |
|  | 内核管理器 | 数据源监控器 | updateIoDevice | 更新IoDevice | IIoDeviceDao.updateIoDevice() | IoDevice | 需要更新的IoDevice对象 |
|  | 内核管理器 | 数据源监控器 | updateVariableTag | 更新数据点 | IVariableTagDao.updateVariableTag() | VariableTag | 需要更新的VariableTag对象 |
|  | 内核管理器 | 数据源监控器 | getDataPoint | 查询数据点 | IVariableTagDao.getDataPoint() | String | 查询数据点的ID |
|  | 内核管理器 | 数据发布器 | getPublisher | 查询Publisher | PublisherDao.getPublisher() | String | 查询Publisher的ID |
|  | 内核管理器 | 数据源监控器 | deleteIoDevice | 删除IoDevice | IIoDeviceDao.deleteIoDevice() | String | 删除IoDevice的ID |
|  | 内核管理器 | 数据发布器 | deletePublisher | 删除Publisher | PublisherDao.deletePublisher() | String | 删除Publisher的ID |
|  | 内核管理器 | 数据源监控器 | deleteVariableTag | 删除数据点 | IVariableTagDao.deleteVariableTag() | String | 删除数据点的ID |
|  | 数据源监控器 | 内核管理器 | loadModule | 加载模块 | LoadModuleCommon.loadModule() | String | 加载模块的名称 |
| List<ModuleElementDefinition> | 模块所包含的definitions |
|  | 数据发布器 | 内核管理器 | uninstallModule | 卸载模块 | LoadModuleCommon.uninstallModule() | Module | 需卸载的Module对象 |

##### 内部接口

表 ‑5 内核管理器系统内部接口表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 数据源管理器 | 事件触发器 | raiseEvent | 产生事件 | EventManager.raiseEvent() | EventType | 事件类型 |
| long | 事件时间 |
| boolean | 是否可返回普通 |
| int | 事件报警级别 |
| TranslatableMessage | 事件消息描述 |
| Map<String, Object> | 事件上下文 |
|  | 发布器管理器 | 事件触发器 | raiseEvent | 产生事件 | EventManager.raiseEvent() | EventType | 事件类型 |
| long | 事件时间 |
| boolean | 是否可返回普通 |
| int | 事件报警级别 |
| TranslatableMessage | 事件消息描述 |
| Map<String, Object> | 事件上下文 |
|  | 数据源管理器 | 数据点监听者管理器 | addDataPointListener | 添加数据点监听者 | RuntimeManager.addDataPointListener() | String | 数据点的ID |
| DataPointListener | 需添加的数据点监听者 |
|  | 数据源管理器 | 数据点监听者管理器 | removeDataPointListener | 移除数据点监听者 | RuntimeManager.removeDataPointListener() | String | 数据点的ID |
| DataPointListener | 需删除的数据点监听者 |
|  | 数据源管理器 | 数据点监听者管理器 | getDataPointListeners | 获得数据点监听者 | RuntimeManager.getDataPointListeners() | String | 数据点的ID |

#### 业务流程



**图 1‑8 内核管理器业务流程**

内核管理器的核心业务流程如上图所示，具体说明为：

1. 在系统启动的时候，先进行各个模块的加载；
2. 模块加载成功之后，对模块具体实例进行初始化，系统启动完成；
3. 用户提交数据源的查询/控制请求，由数据源管理器提交给具体数据源实例，若操作中异常产生，则调用事件触发器产生事件；
4. 数据源管理器接收由数据源采集到的状态信息，并把结果返回给客户端，同时把状态数据传递给其监听者；
5. 数据点监听者接收到状态数据以后，推送给发布器管理器，由其把数据传送给具体发布器实例，再由发布器实例推送给外部系统；
6. 若在数据推送过程中有异常产生，则调用事件触发器产生异常事件。

### 配置工具

#### 软件架构



**图 1‑9 配置工具软件架构**

上图为配置工具软件架构，包含了主要的控制流程和数据流。配置工具主要是作为设计时的配置系统，提供模型配置服务，并且提供了一层开放的rest服务接口。配置工具主要配置的模型主要包括：监控模型配置、发布器模型配置、报告实模型配置、事件配置、事件处理器配置及系统配置。配置工具的主要流程是通过开放服务接口接收用户配置请求，通过具体配置模块对数据进行配置入库。

#### 系统组成



**图 1‑10 配置工具系统组成**

配置工具系统主要包含的模块如上图所示，具体说明如下：

1. 监控模型配置，负责监控模型的配置，包括Cluster、NetworkAddress、IoServer、Board、Port、IoDevice、VariableTag的配置。
2. 发布器配置，负责数据发布器的配置，其中在配置时需选择其需要发布的数据点对象。
3. 报告实例配置，负责报告实例的配置，在配置时需要选择需要生成报告的数据点。
4. 系统配置，负责系统属性的配置，包括实例描述配置、邮件配置、事件配置、Http相关设置、事件清理设置、UI设置、国际化设置、背景颜色设置。
5. 事件配置，主要负责定时事件、维护事件及数据源和发布器的事件配置。
6. 事件处理器配置，主要根据配置的事件来配置事件的处理方式，目前处理方式包括指令处理方式、设置点值处理方式及发送Email处理方式。
7. 数据库，负责存储设计时的模型对象。

#### 接口关系

##### 外部接口

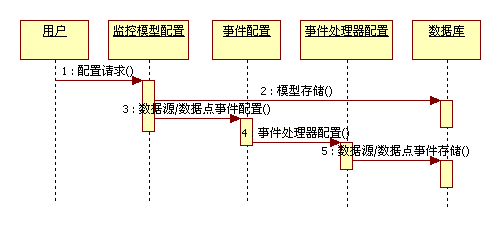
表 ‑6 配置工具外部接口表

| 序号 | 分类 | 类型 | 服务名 | 说明 | 备注 |
| --- | --- | --- | --- | --- | --- |
|  | 集群模型配置接口 | 写入接口 | cluster/create | 创建Cluster对象 |  |
|  | 写入接口 | cluster/delete/{id} | 根据ID删除Cluster对象 |  |
|  | 写入接口 | cluster/update | 更新Cluster对象 |  |
|  | 读取接口 | cluster/find/{id} | 根据ID查询Cluster对象 |  |
|  | 读取接口 | cluster/findall | 查询所有的Cluster对象 |  |
|  | 写入接口 | networkaddress/create | 创建NetworkAddress对象 |  |
|  | 写入接口 | networkaddress/delete/{id} | 根据ID删除NetworkAddress对象 |  |
|  | 写入接口 | networkaddress/update | 更新NetworkAddress对象 |  |
|  | 读取接口 | networkaddress/find/{id} | 根据ID查询NetworkAddress对象 |  |
|  | 读取接口 | networkaddress/findall | 查询所有的NetworkAddress对象 |  |
|  | 写入接口 | ioserver/create | 创建IoServer对象 |  |
|  | 写入接口 | ioserver/delete/{id} | 根据ID删除IoServer对象 |  |
|  | 写入接口 | ioserver/update | 更新IoServer对象 |  |
|  | 读取接口 | ioserver/find/{id} | 根据ID查询IoServer对象 |  |
|  | 读取接口 | ioserver/findall | 查询所有的IoServer对象 |  |
|  | 写入接口 | reportserver/create | 创建ReportServer对象 |  |
|  | 写入接口 | reportserver/delete/{id} | 根据ID删除ReportServer对象 |  |
|  | 写入接口 | reportserver/update | 更新ReportServer对象 |  |
|  | 读取接口 | reportserver/find/{id} | 根据ID查询ReportServer对象 |  |
|  | 读取接口 | reportserver/findall | 查询所有的ReportServer对象 |  |
|  | 写入接口 | trendserver/create | 创建TrendServer对象 |  |
|  | 写入接口 | trendserver/delete/{id} | 根据ID删除TrendServer对象 |  |
|  | 写入接口 | trendserver/update | 更新TrendServer对象 |  |
|  | 读取接口 | trendserver/find/{id} | 根据ID查询TrendServer对象 |  |
|  | 读取接口 | trendserver/findall | 查询所有的TrendServer对象 |  |
|  | 写入接口 | alarmserver/create | 创建AlarmServer对象 |  |
|  | 写入接口 | alarmserver/delete/{id} | 根据ID删除AlarmServer对象 |  |
|  | 写入接口 | alarmserver/update | 更新AlarmServer对象 |  |
|  | 读取接口 | alarmserver/find/{id} | 根据ID查询AlarmServer对象 |  |
|  | 读取接口 | alarmserver/findall | 查询所有的AlarmServer对象 |  |
|  | 写入接口 | board/create | 创建Board对象 |  |
|  | 写入接口 | board/delete/{id} | 根据ID删除Board对象 |  |
|  | 写入接口 | board/update | 更新Board对象 |  |
|  | 读取接口 | board/find/{id} | 根据ID查询Board对象 |  |
|  | 读取接口 | board/findall | 查询所有的Board对象 |  |
|  | 写入接口 | port/create | 创建Port对象 |  |
|  | 写入接口 | port/delete/{id} | 根据ID删除Port对象 |  |
|  | 写入接口 | port/update | 更新Port对象 |  |
|  | 读取接口 | port/find/{id} | 根据ID查询Port对象 |  |
|  | 读取接口 | port/findall | 查询所有的Port对象 |  |
|  | 写入接口 | iodevice/create | 创建IoDevice对象 |  |
|  | 写入接口 | iodevice/update | 更新IoDevice对象 |  |
|  | 读取接口 | iodevice/find/{id} | 根据ID查询IoDevice对象 |  |
|  | 读取接口 | iodevice/findbyprotocol/{protocol} | 根据协议类型查询IoDevice对象 |  |
|  | 读取接口 | iodevice/findbyxid/{xid} | 根据XID查询IoDevice对象 |  |
|  | 读取接口 | iodevice/findids/{protocol} | 根据协议类型获的所有设备ID |  |
|  | 读取接口 | iodevice/findall | 查询所有的IoDevice |  |
|  | 写入接口 | iodevice/delete/{id} | 根据ID删除IoDevice |  |
|  | 写入接口 | iodevice/deletebyprotocol/{protocol} | 根据协议删除IoDevice |  |
|  | 写入接口 | command/create | 创建Command对象 |  |
|  | 写入接口 | command/delete/{id} | 根据ID删除Command对象 |  |
|  | 写入接口 | command/update | 更新Command对象 |  |
|  | 读取接口 | command/find/{id} | 根据ID查询Command对象 |  |
|  | 读取接口 | command/findall | 查询所有的Command对象 |  |
|  | 写入接口 | variabletag/create | 创建VariableTag对象 |  |
|  | 写入接口 | variabletag/delete/{id} | 根据ID删除VariableTag |  |
|  | 写入接口 | variabletag/deletebyiodeviceid/{ioDeviceId} | 根据IoDevice的ID删除所包含的VariableTags |  |
|  | 写入接口 | variabletag/update | 更新VariableTag对象 |  |
|  | 读取接口 | variabletag/find/{id} | 根据ID查询VariableTag |  |
|  | 读取接口 | variabletag/findbydatatypes | 根据数据类型查询VariableTag |  |
|  | 读取接口 | variabletag/findall | 查询所有的VariableTag对象 |  |
|  | 读取接口 | variabletag/findbyiodeviceid/{ioDeviceId} | 根据IoDevice的ID查询所包含的VariableTags |  |
|  | 读取接口 | variabletag/findbyxid/{xid} | 根据XID查询VariableTag |  |
|  | 读取接口 | variabletag/findids/{ioDeviceId} | 根据IoDevice的ID查询其所包含的VariableTags的IDS |  |
|  | 写入接口 | variabletag/textrenderer | 设置VariableTag的文本格式 |  |
|  | 写入接口 | trendtag/create | 创建TrendTag对象 |  |
|  | 写入接口 | trendtag/delete/{id} | 根据ID删除TrendTag对象 |  |
|  | 写入接口 | trendtag/update | 更新TrendTag对象 |  |
|  | 读取接口 | trendtag/find/{id} | 根据ID查询TrendTag对象 |  |
|  | 读取接口 | trendtag/findall | 查询所有的TrendTag对象 |  |
|  | 邮件接口 | 读取接口 | mailinglist/getmailinglists\_all | 查询所有的邮件 |  |
|  | 读取接口 | mailinglist/getmailinglist/{id} | 根据ID获取邮件对象 |  |
|  | 读取接口 | mailinglist/getmailinglist\_byxid/{xid} | 根据XID获取邮件对象 |  |
|  | 写入接口 | mailinglist/savemailinglist | 保存邮件对象 |  |
|  | 写入接口 | mailinglist/deletemailingList/{id} | 根据ID删除邮件对象 |  |
|  | 获取接口 | mailinglist/generate\_unique\_xid | 产生唯一XID |  |
|  | 点连接接口 | 获取接口 | pointlinkcontroller/getsourcevartags | 获取源数据点 |  |
|  | 获取接口 | pointlinkcontroller/gettargetvartags | 获取目标数据点 |  |
|  | 获取接口 | pointlinkcontroller/getpointlinks | 获取所有的点连接 |  |
|  | 获取接口 | pointlinkcontroller/getpointlink/{id} | 根据ID获取点连接 |  |
|  | 写入接口 | pointlinkcontroller/create | 创建点连接 |  |
|  | 写入接口 | pointlinkcontroller/delete/{id} | 根据ID删除点连接 |  |
|  | 写入接口 | pointlinkcontroller/validatescript | 校验script |  |
|  | 发布器接口 | 获取接口 | publisher/getpublishertypes | 获取publisher的所有的类型 |  |
|  | 获取接口 | publisher/getpublishers | 获取所有的publisher |  |
|  | 获取接口 | publisher/getpublishervo | 获取publisher设计时对象 |  |
|  | 系统配置接口 | 读取接口 | systemsettings/getinstancedescsetting | 获得实例描述设置 |  |
|  | 读取接口 | systemsettings/getemailsettings | 获得邮件相关设置 |  |
|  | 读取接口 | systemsettings/geteventtypesettings | 获得事件类型设置 |  |
|  | 读取接口 | systemsettings/gethttpsettings | 获得Http相关设置 |  |
|  | 读取接口 | systemsettings/geteventpurgesettings | 获得事件清理设置 |  |
|  | 读取接口 | systemsettings/getuiperformancesetting | 获得UI设置 |  |
|  | 读取接口 | systemsettings/getfuturedatelimitperiod | 获得时间周期设置 |  |
|  | 读取接口 | systemsettings/getlanguage | 获得语言设置 |  |
|  | 读取接口 | systemsettings/getcolour | 获得颜色相关设置 |  |
|  | 读取接口 | systemsettings/getdatabasesize | 获得数据库目录大小 |  |
|  | 读取接口 | systemsettings/getfiledatacount | 获得所有文件数量 |  |
|  | 读取接口 | systemsettings/getfiledatasize | 获得所有文件大小 |  |
|  | 读取接口 | systemsettings/getpointhistorycountlist | 获得历史点数量对象的list |  |
|  | 读取接口 | systemsettings/getpointhistorycountsum | 获得历史点数量总数 |  |
|  | 读取接口 | systemsettings/geteventcount | 获得事件数量 |  |
|  | 写入接口 | systemsettings/saveemailsettings | 保存邮件设置 |  |
|  | 写入接口 | systemsettings/savesystemeventalarmlevels | 保存系统事件报警级别 |  |
|  | 写入接口 | systemsettings/saveauditeventalarmlevels | 保存审计事件报警级别 |  |
|  | 写入接口 | systemsettings/savehttpsettings | 保存Http设置 |  |
|  | 写入接口 | systemsettings/savemiscsettings | 保存其他设置 |  |
|  | 写入接口 | systemsettings/saveinfosettings | 保存实例设置 |  |
|  | 写入接口 | systemsettings/savecoloursettings | 保存颜色设置 |  |
|  | 写入接口 | systemsettings/savelanguagesettings | 保存语言设置 |  |
|  | 写入接口 | systemsettings/savesettings | 保存所有设置信息 |  |
|  | 读取接口 | systemsettings/getvalue | 根据KEY和默认Value值获取设置Value |  |
|  | 读取接口 | systemsettings/getintvalue/{key} | 根据KEY值获取设置Value |  |
|  | 读取接口 | systemsettings/getintvaluewithdefaultvalue | 根据默认Value获取int Value |  |
|  | 读取接口 | systemsettings/getbooleanvalue/{key} | 根据KEY获取bool类型的Value |  |
|  | 读取接口 | systemsettings/getsystemsettingsdefinitions | 获取所有的SystemSettingsDefinitions |  |
|  | 读取接口 | systemsettings/getdatabasetypename | 获取数据库类型名称 |  |
|  | 事件处理接口 | 读取接口 | eventhandler/geteventtypes | 根据协议类型获得该协议设备的事件类型 |  |
|  | 读取接口 | eventhandler/geteventhandlers | 根据EventTypeVo获取它的处理方式 |  |
|  | 读取接口 | eventhandler/getusereventtypelist | 获取所有UserEventTypeVO实例 |  |
|  | 读取接口 | eventhandler/getsystemevents | 获取所有SystemEventTypeVO实例 |  |
|  | 读取接口 | eventhandler/getauditevents | 获取所有AuditEventTypeVO实例 |  |
|  | 读取接口 | eventhandler/getadmineventtypes | 获取所有AdminEventTypeVO实例 |  |
|  | 写入接口 | eventhandler/savesetpointeventhandler | 保存点事件处理实例 |  |
|  | 写入接口 | eventhandler/saveemaileventhandler | 保存邮件事件处理实例 |  |
|  | 写入接口 | eventhandler/saveprocesseventhandler | 保存处理事件处理实例 |  |
|  | 写入接口 | eventhandler/delete/{id} | 根据ID删除事件处理实例 |  |
|  | Graphical Views接口 | 写入接口 | graphicalviews/saveview | 保存View |  |
|  | 读取接口 | graphicalviews/view/{id} | 获取指定ID的GraphicalView，并验证用户是否具有相关权限 |  |
|  | 读取接口 | graphicalviews/viewbyxid/{xid} | 获取指定xid的GraphicalView，并验证用户是否具有相关权限 |  |
|  | 读取接口 | graphicalviews/viewbyname/{name} | 获取指定name的GraphicalView，并验证用户是否具有相关权限 |  |
|  | 写入接口 | graphicalviews/delete/{id} | 删除View |  |
|  | 事件维护接口 | 读取接口 | maintenanceevent/findall | 获取所有的事件维护实例 |  |
|  | 读取接口 | maintenanceevent/find/{id} | 通过ID查询事件维护实例 |  |
|  | 写入接口 | maintenanceevent/create | 创建事件维护实例 |  |
|  | 写入接口 | maintenanceevent/delete/{id} | 根据ID删除事件维护实例 |  |
|  | 读取接口 | maintenanceevent/generateuniquexid | 获取一个唯一的XID |  |
|  | 定时事件接口 | 读取接口 | scheduledevent/findall | 获取所有的定时事件 |  |
|  | 读取接口 | scheduledevent/find/{id} | 通过ID获取定时事件 |  |
|  | 写入接口 | scheduledevent/create | 创建定时事件 |  |
|  | 写入接口 | scheduledevent/delete/{id} | 根据ID删除定时事件 |  |
|  | 报表接口 | 读取接口 | report/getreports/{userId} | 根据USERID获取该用户下的ReportVO |  |
|  | 读取接口 | report/getreport | 根据ID获取ReportVO |  |
|  | 写入接口 | report/savereport | 添加ReportVO |  |
|  | 写入接口 | report/deletereport/{id} | 根据ID删除ReportVO对象 |  |
|  | 本地变量处理接口 | 写入接口 | localvariabletag/create | 创建本地变量 |  |
|  | 写入接口 | localvariabletag/delete/{id} | 根据ID删除本地变量 |  |
|  | 写入接口 | localvariabletag/update | 更新本地变量 |  |
|  | 读取接口 | localvariabletag/find/{id} | 根据ID获取本地变量 |  |
|  | 读取接口 | localvariabletag/findall | 获取所有的本地变量 |  |
|  | 读取接口 | localvariabletag/generatexid | 生成唯一XID |  |

##### 内部接口

无

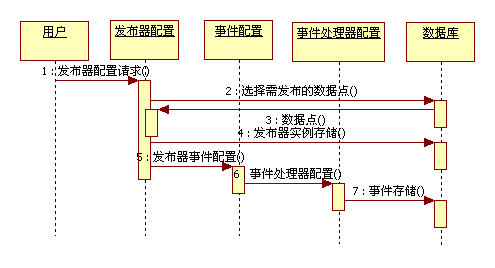
#### 业务流程



**图 1‑11 监控模型配置业务流程**

配置监控模型的业务流程如上图所示，主要步骤包括：

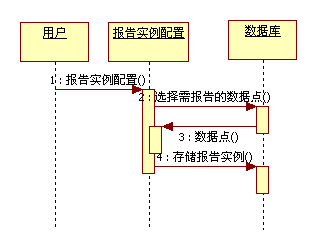
1. 用户提交监控模型配置请求；
2. 监控模型配置接收用户的请求，根据参数生成模型对象，并进行存储；
3. 配置完监控模型，选择配置对象，添加数据源/数据点的事件配置；
4. 由事件配置模型进行配置，并交由事件处理器配置对事件的处理方式进行配置，最后存储入库。

****

**图 1‑12 发布器模型配置业务流程**

发布器模型配置业务流程如上图所示，其主要步骤包括：

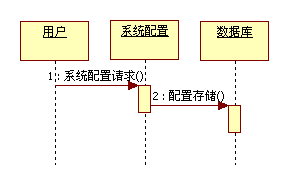
1. 用户提交发布器配置请求；
2. 发布器配置中选择需要发布的数据点，存储发布器实例；
3. 模型配置完成后，通过事件配置模块进行事件的配置；
4. 通过事件处理器配置发布器事件的处理方式，最后存储入库。

****

**图 1‑13 报告实例配置业务流程**

报告实例配置的业务流程如上图所示，其主要步骤包括：

1. 用户提交报告实例配置请求，并选择需报告的数据点；
2. 报告实例配置模块进行报告实例的生产及存储。

****

**图 1‑14 系统配置业务流程**

系统配置的主要流程如上图所示，主要步骤为：

1. 用户选择需配置的属性，提交配置请求；
2. 系统配置模块接收到请求之后，进行配置存储入库。

### 数据源监控器

#### 软件架构

**图 1‑15 数据源监控器软件架构**

数据源监控器是应用网关的核心技术，它提供的关键流程包括：运行时实例的加载、数据源运行时实例初始化、外部数据源状态查询/控制。首先，数据源模块接收内核管理器控制请求，并根据具体数据源类型进行参数适配，这时，再创建运行时实例及初始化数据源建立通信连接，这样数据源就可以正常运行了；然后，再由运行时数据源实例接收内核管理器传递的查询/控制数据点请求，交由通过通信连接获取数据点的状态，这时运行时数据源把获取的状态信息存储入库，并提供给数据分析服务系统及内核管理器系统使用。

#### 系统组成



**图 1‑16 数据源监控器系统组成**

数据源监控器的主要系统组成如上图所示，主要由以下几个模块组成：

1. 轮询数据源模块

轮询数据源模块负责主动型数据源的模型管理，它对主动型属于模型进行了定义及设计时的管理。

1. 事件数据源模块

事件数据源模块负责事件型数据源的模型管理，事件数据源需要等待事件的触发才会对外部数据源发送查询/控制请求，该模板主要是对其模型定义及设计时管理。

1. 数据源参数适配器

数据源参数适配器是为了实现不同协议的数据源能进行统一的管理，对不同协议数据源的参数进行适配。

1. 运行时数据源

运行时数据源定义了数据源运行时的一些属性及行为，它负责触发数据源的通信连接及数据源发送查询/控制令，并对响应结果的处理及把变化信息告诉应用网关其他系统。

1. 数据源初始化连接

数据源初始化连接负责与外部数据源的通信连接，它会根据具体不同协议的通信类型建立其通信服务，通常类型有：TCP/IP、串口及HTTP等。

1. 数据源状态轮询

数据源状态轮询负责数据源的状态查询操作，它负责不同协议数据源的查询令发送。

1. 数据源状态控制

数据源状态控制负责数据源的控制令操作，它负责对控制令的编码及响应的解码。

1. 数据库

数据库负责存储数据点的状态信息。

#### 接口关系

##### 外部接口

表 ‑7 数据源监控器外部接口表

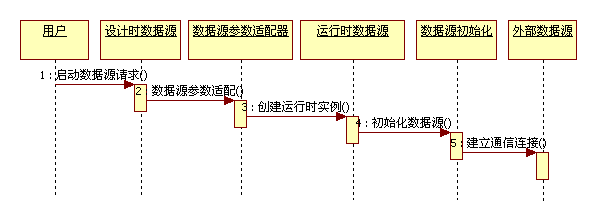
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 内核管理器 | 数据源监控器 | initialize | 启动数据点 | IDataPointRT.initialize() | 无 | 无 |
|  | 内核管理器 | 数据源监控器 | beginPolling | 开始数据源轮询 | IDataSourceRT.beginPolling() | 无 | 无 |
|  | 内核管理器 | 数据源监控器 | terminate | 停止数据点 | IDataPointRT.terminate() | 无 | 无 |
|  | 内核管理器 | 数据源监控器 | terminate | 停止数据源 | IDataSourceRT.terminate() | 无 | 无 |
|  | 数据发布器 | 数据源监控器 | setPointValue | 设置数据点值 | IDataSourceRT.setPointValue() | IDataPointRT | 需要设置值的数据点对象 |
| PointValueTime | 数据点值 |
| SetPointSource | 设置点源头 |
|  | 数据发布器 | 数据源监控器 | setBatchPointValues | 多点控制 | IDataSourceRT.setBatchPointValues() | Map<IDataPointRT, PointValueTime> | 多点组成的Map |
| SetPointSource | 设置点值源头 |

##### 内部接口

表 ‑8 数据源监控器内部接口表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 数据源模块 | 数据源参数适配器 | convertParameters | 数据源参数适配 | ParametersConverter.convertParameters() | IoDevice | 需要参数适配的IoDevice |
| ParameterConvertBean | 参数转换生成的bean |
|  | 运行时数据源模块 | 数据源模块 | createDataSourceRT | 创建运行时数据源实例 | DataSourceDefinition.createDataSourceRT() | IoDevice | 数据源对象 |
|  | 运行发布器实例 | 数据源初始化连接 | initialize | 初始化与外部数据源连接 | IDataSourceRT.initialize() | 无 | 无 |
|  | 运行发布器实例 | 查询/控制模块 | beginPolling | 状态进行轮询 | IDataSourceRT.beginPolling() | 无 | 无 |
|  | 运行发布器实例 | 查询/控制模块 | setPointValue | 设置数据点值 | IDataSourceRT.setPointValue() | IDataPointRT | 需要设置值的数据点对象 |
| PointValueTime | 数据点值 |
| SetPointSource | 设置点源头 |
|  | 运行发布器实例 | 查询/控制模块 | setBatchPointValues | 多点控制 | IDataSourceRT.setBatchPointValues() | Map<IDataPointRT, PointValueTime> | 多点组成的Map |
| SetPointSource | 设置点值源头 |
|  | 运行发布器实例 | 数据库 | savePointValueSync | 同步更新数据点 | PointValueDao.savePointValueSync() | String | 数据点ID |
| PointValueTime | 数据点的值 |
| SetPointSource | 数据点值更新源 |
|  | 运行发布器实例 | 数据库 | savePointValueAsync | 异步更新数据点 | PointValueDao.savePointValueAsync() | String | 数据点ID |
| PointValueTime | 数据点的值 |
| SetPointSource | 数据点值更新源 |

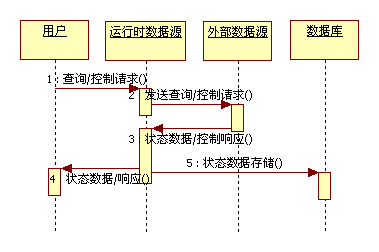
#### 业务流程



**图 1‑17 数据源启动业务流程**

数据源启动业务的流程如上图所示，具体操作步骤如下所示：

1. 用户提交启动数据源请求；
2. 设计时数据源模块调用数据源参数适配器进行参数适配；
3. 由设计时模型创建运行时实例，再由运行时实例进行初始化数据源；
4. 数据源实例初始化并建立通信连接。



**图 1‑18 发送查询/控制业务流程**

发送查询/控制业务的流程如上图所示，具体操作步骤如下所示：

1. 用户编辑并提交查询/控制请求；
2. 运行时数据源对查询/控制请求进行解析，并把查询令/控制令发送给外部数据源；
3. 运行时数据源接收外部数据源返回状态数据/控制响应；
4. 运行时数据源把结果返回给用户，并对其进行数据存储。

### 数据发布器

#### 软件架构



**图 1‑19 数据发布器软件架构**

数据发布器系统建立了应用网关与外部系统的数据交互通道及为多级监控提供了有力的保障，它的软件架构如上图所示。发布器的类型分为主动型发布器和被动型发布器，主动型发布器是数据发布器作为Client端，当有数据点值更新或传递周期到了，发布器主动把数据发送到外部系统会上级系统；被动型发布器是作为Server端，它等待外部系统建立连接及数据请求。数据发布器系统还提供了可扩展接口，可以实现不同类型发布器。

#### 系统组成



**图 1‑20 数据发布器系统组成**

数据发布器的系统组成如上图所示，主要包括以下几个模块：

1. 发布器实例管理

发布器实例管理负责发布器设计时和运行时实例的统一管理，并负责运行时发布器实例的创建，同时，负责接收外部系统的控制请求，并根据请求事项进行相应处理。

1. 数据点监听模块

数据点监听模块主要是负责监听数据点值的更新或变化情况，若有更新或变化，把数据点值传递给具体运行时发布器实例。

1. 运行时发布器实例

发布器运行时实例负责发布器的初始化及数据的推送及接收。

1. 发布器初始化连接

发布器初始化连接负责数据发布器与外部系统或上级系统建立通信连接，连接方式有TCP/IP、HTTP等方式。

1. 主动发布器

主动发布器负责主动与外部系统或上级系统，把状态数据主动推送给外部系统或上级系统。

1. 被动发布器

被动发布器负责监听外部系统或上级系统的连接，当有连接的时候，把监听到的状态数据推送给上级系统。

#### 接口关系

##### 外部接口

表 ‑9 数据发布器外部接口表

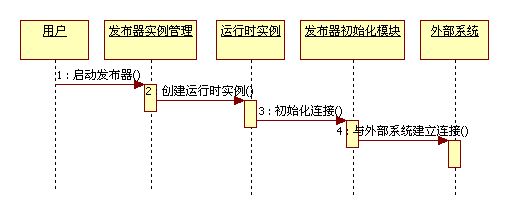
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 内核管理器 | 数据发布器 | initialize | 初始化发布器 | IPublisherRT.initialize() | 无 | 无 |
|  | 内核管理器 | 数据发布器 | terminate | 停止发布器 | IPublisherRT.terminate() | 无 | 无 |
|  | 内核管理器 | 数据发布器 | pointUpdated | 监听数据点值更新 | DataPointListener.pointUpdated() | PointValueTime | 数据点的新值 |
|  | 内核管理器 | 数据发布器 | pointChanged | 监听数据点值改变 | DataPointListener.pointChanged() | PointValueTime | 数据点的旧值 |
| PointValueTime | 数据点的新值 |
|  | 数据发布器 | 外部系统 | initialize | 与外部系统建立连接 | SendThread.initialize() | 无 | 无 |
|  | 数据发布器 | 外部系统 | run | 推送数据给外部系统 | SendThread.run() | 无 | 无 |

##### 内部接口

表 ‑10 数据发布器内部接口表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 运行发布器实例 | 发布器实例管理 | createPublisherRT | 创建运行时发布器实例 | PersistentSenderVO.createPublisherRT() | PublisherVO | 发布器设计时对象 |
|  | 运行发布器实例 | 发布时初始化连接 | initialize | 停止发布器 | IPublisherRT.initialize() | 无 | 无 |
|  | 运行发布器实例 | 发送数据模块 | pointUpdated | 监听数据点值更新 | DataPointListener.pointUpdated() | PointValueTime | 数据点的新值 |
|  | 运行发布器实例 | 发送数据模块 | runImpl | 监听数据点值改变 | SendThread.runImpl() | 无 | 无 |
|  | 数据点监听模块 | 运行发布器实例 | publish | 运行发布器发布新值 | PublisherRT.publish() | PublishedPointVO | 发布器数据点 |
| PointValueTime | 更新的点的值 |

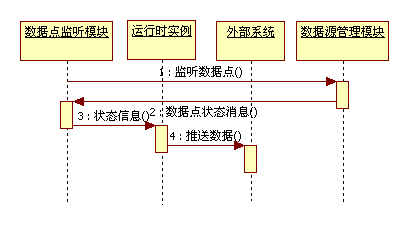
#### 业务流程



**图 1‑21 数据发布器启动业务流程**

数据发布器启动的业务流程如上图所示，具体步骤如下：

1. 用户提交启动发布器请求；
2. 发布器实例管理模块接收请求，根据实例创建运行时实例；
3. 运行时发布器实例初始化通信连接，与外部系统或上级系统建立连接。



**图 1‑22 数据发布器发布数据业务流程**

数据发布器发布业务数据的流程如上图所示，具体步骤如下：

1. 由数据点监听模块监听数据点值的更新状态；
2. 若有数据点更新，获取数据点状态；
3. 状态信息由运行时实例推送给外部系统或上级系统。

### 事件管理器

#### 软件架构



**图 1‑23 事件管理器软件架构**

事件管理器系统负责整个应用网关的事件服务。其主要负责事件的发布、存储、处理、事件类型扩展及事件处理方式扩展等功能。其他服务系统提交发布事件请求到事件管理器模块，事件管理器根据事件请求选择合适的事件产生器，如定时事件、系统事件、维护事件等，在事件产生之后，调用其配置的事件处理方式进行事件的处理，其中处理方式常用的有指令处理、点设置处理及发送Email处理。同时，在软件架构中，定义了事件类型及事件处理方式的扩展接口。

#### 系统组成



**图 1‑24 事件管理器系统组成**

事件管理器系统组成如上图所示，它的组成模块主要有：

1. 事件管理器

事件处理器是事件管理器的枢纽模块，它负责建立其他子系统与事件管理器的数据通道。它的主要实现接收其他子系统的事件请求，包括事件的产生、事件实例的查询、事件的取消及事件的处理请求等。

1. 定时事件

定时事件是一种扩展事件类型，定义了一个定时器，其定时周期可以是年、月、天、小时、分，当定时器触发的时候产生定时器事件，它主要负责定时事件的定义及产生。

1. 系统事件

系统事件是网关系统拥有的基础事件，它定义了数据源事件、数据点事件、审计事件及基本事件，它主要负责该些类型事件的产生及其处理器的管理。

1. 维护事件

维护事件是一种扩展事件，它主要是负责数据源事件的维护，维护方式分为：自动和手动，若设置为自动维护，当事件产生的时候，就自动对其进行确认。

1. 事件处理器

事件处理器定义了系统中事件的处理方式，处理方式有指令处理、点设置处理及发送Email处理。当事件触发时，会调用其配置的处理方式进行相应操作（同一事件可以同时配置多种处理方式）。同时，事件处理方式是可扩展的。

1. 数据库

数据库主要负责事件数据的存储。

1. Dao模块

Dao模块负责事件实例的增删改成操作。

#### 接口关系

##### 外部接口

表 ‑11 事件管理器外部接口表

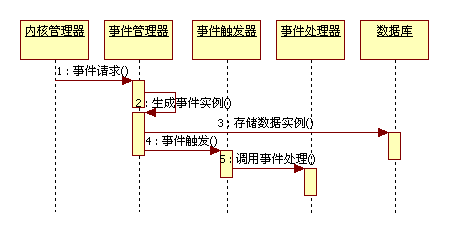
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 内核管理器 | 事件管理器 | raiseEvent | 产生事件 | SystemEventType.raiseEvent() | EventType | 事件类型 |
| long | 事件时间 |
| boolean | 是否可返回普通 |
| TranslatableMessage | 事件消息描述 |
|  | 内核管理器 | 事件管理器 | returnToNormal | 事件变为普通 | EventInstance.returnToNormal() | long | 时间 |
| int | 事件变为普通的原因 |

##### 内部接口

表 ‑12 事件管理器内部接口表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 调用方 | 服务方 | 接口名称 | 用途 | 调用方式 | 参数类型 | 参数意义 |
|  | 事件管理器 | Dao模块 | saveEvent | 存储事件 | EventDao.saveEvent() | EventInstance | 需要存储的事件 |
|  | 事件管理器 | Dao模块 | insertUserEvents | 插入用户事件 | EventDao.insertUserEvents() | String | 事件ID |
| List<String> | 用户IDS |
| boolean | 是否报警 |
|  | 事件管理器 | Dao模块 | ackEvent | 确认事件 | EventDao. ackEvent() | String | 事件ID |
| long | 确认时间 |
| String | 用户ID |
|  | 事件管理器 | Dao模块 | getActiveEvents | 获得处于激活状态的事件 | EventDao.getActiveEvents() | String | 事件类型名称 |
|  | 事件管理器 | Dao模块 | getEventHandlers | 获得事件处理方式 | EventDao.getEventHandlers() | EventType | 事件的类型 |
|  | 事件管理器 | 定时事件 | raiseEvent | 产生定时事件 | ScheduledEventRT.raiseEvent() | long | 产生事件时间 |
|  | 事件管理器 | 定时事件 | returnToNormal | 事件恢复正常 | ScheduledEventRT.returnToNormal() | long | 事件恢复正常时间 |
|  | 事件管理器 | 维护事件 | raiseEvent | 产生维护事件 | MaintenanceEventRT.raiseEvent() | long | 产生事件时间 |
|  | 事件管理器 | 维护事件 | returnToNormal | 事件恢复正常 | MaintenanceEventRT.returnToNormal() | long | 事件恢复正常时间 |
|  | 事件管理器 | 系统事件 | raiseEvent | 产生系统事件 | SystemEventType.raiseEvent() | long | 产生事件时间 |
| boolean | 是否可恢复正常 |
|  | 事件管理器 | 系统事件 | returnToNormal | 事件恢复正常 | SystemEventType.returnToNormal() | long | 事件恢复正常时间 |
|  | 事件模块 | 事件处理器 | eventRaised | 事件产生的处理 | EventHandlerRT.eventRaised() | EventInstance | 需要处理的事件实例 |
|  | 事件模块 | 事件处理器 | eventInactive | 事件不使能的处理 | EventHandlerRT.eventInactive() | EventInstance | 需要处理的事件实例 |

#### 业务流程



**图 1‑25 事件发布流程**

事件发布去流程如上图所示，其主要步骤如下：

1. 内核管理器提交事件请求到事件管理器；
2. 事件管理器首先根据事件类型生成事件实例；
3. 对生成事件实例进行存储；
4. 事件管理器把事件触发消息传递给具体事件产生器；
5. 事件触发器事件产生后，同时调用事件处理服务。

### 数据分析服务

#### 软件架构



**图 1‑26 数据分析服务软件架构**

数据分析服务软件架构如上图所示，数据分析服务主要分为三个方面：报表服务、趋势服务及历史服务。报表服务可以在系统运行时显示在画面上，报表运行后可以打印，或者保存在硬盘上为以后提供打印或显示。趋势服务对数据点值的分析统计，根据趋势模板生成趋势结果图。历史服务获取数据点的历史数据值，可以根据多种关键字进行访问。

#### 系统组成



**图 1‑27 数据分析服务系统组成**

数据分析服务系统组成如上图所示，主要由三个模块组成：

1. 报表服务，是基于报表的一种数据分析，主要提供对数据点值的分析，可以通过不同的分析方法进行分析，分析结果提供文字、数值、图形等多种形式输出。
2. 趋势服务，提供了数据点值的趋势和变化规律的分析，提供了定比和环比分析方法，并提供了多种趋势显示模板。

历史服务，主要是负责数据点的历史数据值服务，可以通过不同关键字进行历史数据检索服务。

#### 接口关系

##### 外部接口

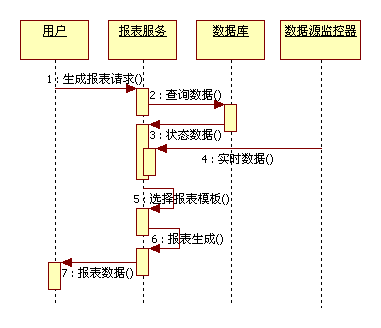
表 ‑13 数据分析服务外部接口表

| 序号 | 分类 | 类型 | 服务名 | 说明 | 备注 |
| --- | --- | --- | --- | --- | --- |
|  | 数据导入导出接口 | 写入接口 | emport/createexportdata | 创建导出数据 |  |
|  | 读取接口 | emport/importdata | 导出数据 |  |
|  | 读取接口 | emport/getimportstatus | 获得ImportTask的状态 |  |
|  | 发送接口 | emport/cancelimport | 取消ImportTask任务 |  |
|  | 读取接口 | emport/getemportdefinitions | 获取所有的EmportDefinitions |  |
|  | 数据点详细信息接口 | 读取接口 | variabletagdetails/getvartagbyxid/{xid} | 通过XID获取VariableTag |  |
|  | 读取接口 | variabletagdetails/getuservariabletags | 获取当前用户权限下的VariableTags |  |
|  | 读取接口 | variabletagdetails/getvartagaccesstype | 获取当前用户对VariableTag的操作权限 |  |
|  | 读取接口 | variabletagdetails/getlatestpointvaluetimes | 获取点的最新值 |  |
|  | 读取接口 | variabletagdetails/getvartagwithsetperm/{id} | 通过ID获取VariableTag，并且用户对其需有set权限 |  |
|  | 读取接口 | variabletagdetails/getvartagwithreadperm/{id} | 通过ID获取VariableTag，并且用户对其需有read权限 |  |
|  | 读取接口 | variabletagdetails/getvartagbyxidwithsetperm/{xid} | 通过XID获取VariableTag，并且用户对其需有set权限 |  |
|  | 读取接口 | variabletagdetails/getvartagbyxidwithreadperm/{xid} | 通过XID获取VariableTag，并且用户对其需有read权限 |  |
|  | 读取接口 | variabletagdetails/getdatapointrt/{id} | 通过ID获取DataPointRT |  |
|  | 写入接口 | variabletagdetails/forcepointread | 通过ID设置VariableTag可读 |  |
|  | 读取接口 | variabletagdetails/generatevartagxid | 产生VariableTag的XID |  |
|  | 读取接口 | variabletagdetails/generateiodevicexid | 产生IoDevice的XID |  |
|  | 报表接口 | 读取接口 | report/getreports/{userId} | 根据USERID获取该用户下的ReportVO |  |
|  | 读取接口 | report/getreport | 根据ID获取ReportVO |  |
|  | 写入接口 | report/savereport | 添加ReportVO |  |
|  | 写入接口 | report/runreport | 运行ReportVO对象 |  |
|  | 写入接口 | report/deletereport/{id} | 根据ID删除ReportVO对象 |  |
|  | 写入接口 | report/deletereportinstance{instanceId} | 根据实例ID删除ReportVO的运行实例 |  |
|  | 读取接口 | report/getreportinstances{userId} | 根据用户ID删除ReportVO的运行实例 |  |
|  | 写入接口 | report/setpreventpurge | ReportVO运行实例设置是否清除数据参数 |  |
|  | 写入接口 | report/createreport\_fromwatchlist | 给WatchList的点创建报表 |  |
|  | 读取接口 | report/getreportinstance | 获取报表运行实例 |  |
|  | Watchlist接口 | 读取接口 | watchlist/get\_watchlist\_pointhierarchy | 获取WatchList的点继承结构 |  |
|  | 写入接口 | watchlist/preparewatchlist/{watchListId} | WatchList包含的Points属性设置为可设置的 |  |
|  | 读取接口 | watchlist/getselectedwatchlist/{userId} | 获得用户选择的WatchList |  |
|  | 读取接口 | watchlist/getwatchlists/{userId} | 获取用户所拥有的WatchLists |  |
|  | 写入接口 | watchlist/createnewwatchlist | 创建一个WatchList |  |
|  | 写入接口 | watchlist/savewatchlist | 存储一个WatchList |  |
|  | 写入接口 | watchlist/saveselectedwatchlist | 保存用户选择的WatchList |  |
|  | 写入接口 | watchlist/updatewatchlistname | 更新WatchList的name |  |
|  | 写入接口 | watchlist/addnewwatchlist | 添加一个新的WatchList |  |
|  | 写入接口 | watchlist/deletewatchlist/{watchListId} | 根据ID删除WatchList |  |
|  | 写入接口 | watchlist/setselectedwatchlist/{watchListId} | 设置选择WatchList中所有点为可设置的 |  |
|  | 读取接口 | watchlist/getwatchlistpointids/{watchListId} | 获取WatchList所有点的ID |  |
|  | 读取接口 | watchlist/getwatchlistpoints/{watchListId} | 获取WatchList的所有点 |  |
|  | 写入接口 | watchlist/addtowatchlist | 添加点到WatchList中 |  |
|  | 写入接口 | watchlist/addupdateshareduser | WatchList中添加SharedUser |  |
|  | 写入接口 | watchlist/removeshareduser | 删除WatchList中的SharedUser |  |
|  | 写入接口 | watchlist/removefromwatchlist | 从WatchList中删除VariableTag |  |
|  | 写入接口 | watchlist/moveup | WatchList中指定VariableTag上移 |  |
|  | 写入接口 | watchlist/movedown | WatchList中指定VariableTag下移 |  |
|  | 读取接口 | watchlist/getwatchlistbyid/{watchListId} | 根据WatchList的ID获得WatchList |  |

##### 内部接口

无

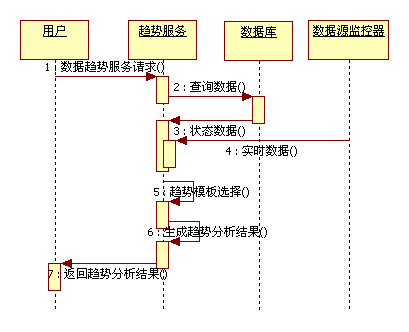
#### 业务流程



**图 1‑28 报表数据服务业务流程**

报表数据服务的业务流程如上图所示，主要步骤如下所示：

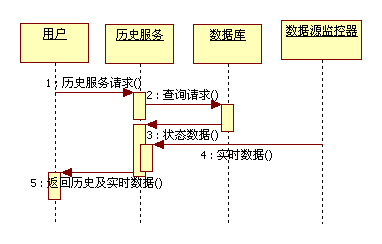
1. 用户提交生成报表请求；
2. 报表服务负责数据点数据的查询，并同时接收数据源监控器的实时数据；
3. 报表服务选择报表模板，根据选择分析方法进行报表分析；
4. 报表的生成，并把报表结果返回给用户。



**图 1‑29 趋势数据服务业务流程**

趋势数据服务的业务流程如上图所示，主要步骤如下所示：

1. 用户生成并提交数据趋势服务请求；
2. 趋势服务接收用户请求，查询数据库数据及获得监控实时数据；
3. 趋势服务选择趋势分析方法及趋势显示模板；
4. 生成趋势分析结果，并返回趋势分析结果。

****

**图 1‑30 历史数据服务业务流程**

历史数据服务的业务流程如上图所示，主要步骤如下所示：

1. 用户生成及提交历史服务请求；
2. 历史服务首先解析用户请求，并根据请求进行数据查询，同时获取数据的实时数据；
3. 历史服务把获取的状态数据返回用户。

## 关键技术

关键技术的描述，要特性几个要素，**必要性(或技术研究所要解决的具体问题)、研究内容(研究×××模型、方法、理论、技术等)、解决什么问题、达到什么效果**。具体参考描述方法，参见软件产品技术方案撰写方法。

## 关键技术解决途径

技术途径或技术解决途径或技术方案，与关键技术和研究内容的关系是，用于说明如何开展研究内容的研究工作，以实现所述关键技术，也就是说，用什么方法、步骤、算法、流程、架构等来开展研究工作，解决问题。

对于一个项目而言，多个关键技术之间必定存在相互关系，在描述技术途径时，可以用图的方式，对各个关键技术间的关系进行表述。同理，对于软件产品的关键技术也是这样。

技术途径的要素，一般可包括：针对某一关键技术或研究内容的细粒度的软件架构图、应用模式图、模型图(数据模型、规范标准模型等)、关键算法描述、方法步骤或流程。

技术途径的描述方式，多种多样，根据解决的技术类型而定，目前，还抽象不出可复用的模式。

## 主要创新点

在关键技术、研究内容、和技术途径的基础上，突出牛逼之处，例如：

1. 提出了××××
2. 达到了××××
3. 满足了××××
4. 填补了××××
5. 实现了××××效果
6. 具备×××××特征
7. 优化了××××
8. 提高了××××
9. 减低了××××
10. 达到××××目的/目标/效果

## 技术指标

## 部署方式

对于软件产品而言，软件实体(子系统、模块之类)是研制成果的载体，一个完整的软件服务或系统，如何部署到硬件平台上，包括网络拓扑关系。

以分布式归档服务为例，需要部署归档管理服务和归档实例服务，且必须描述归档服务系统与存储系统(Hdfs、盘阵、关系数据库)间的拓扑关系。