文档编号：

类 别： 技术

密 级：

**智能插座SP20**

**云平台概要设计说明书**

编 写：

编写时间： 2021年04月

部门名称： 成都研发

审 核：

**猿人创新成都研发**

**2021年04月**

**修订记录**

A–增加 M–修改 D–删除

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 日期 | AMD | 修订内容 | 修订人 | 审核人 |
| V0.1 | 2021.04.26 | A | 初稿 |  | 水熊 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目录**

[云平台概要设计说明书 6](#_Toc73441474)

[1 引言 6](#_Toc73441475)

[1.1 编写背景 6](#_Toc73441476)

[1.2 编写目的 6](#_Toc73441477)

[1.3 定义 7](#_Toc73441478)

[1.4 参考规范 7](#_Toc73441479)

[2 系统架构设计 7](#_Toc73441480)

[2.1 架构设计 7](#_Toc73441481)

[2.1.1 IOT-CLOUD 7](#_Toc73441482)

[2.1.2 数据交互 8](#_Toc73441483)

[2.1.3 微服务 9](#_Toc73441484)

[2.2 关键技术/设计难点/解决方案 9](#_Toc73441485)

[2.3 开发环境配置 10](#_Toc73441486)

[2.4 测试环境配置 10](#_Toc73441487)

[2.5 生产环境配置 10](#_Toc73441488)

[3 数据库设计 11](#_Toc73441489)

[3.1 数据库环境说明 11](#_Toc73441490)

[3.2 数据库命名规则 11](#_Toc73441491)

[3.3 表汇总 11](#_Toc73441492)

[3.4 安全性设计 12](#_Toc73441493)

[3.4.1 防止用户直接操作数据库的方法 12](#_Toc73441494)

[3.4.2 用户账号密码的加密方法 12](#_Toc73441495)

[3.4.3 数据库管理与维护说明 12](#_Toc73441496)

[4 功能设计 12](#_Toc73441497)

[4.1 功能汇总 12](#_Toc73441498)

[4.2 http服务 13](#_Toc73441499)

[4.2.1 功能描述 13](#_Toc73441500)

[4.2.2 功能设计/流程图 13](#_Toc73441501)

[4.2.3 对外接口 15](#_Toc73441502)

[4.2.4 具体实现核心概述 16](#_Toc73441503)

[4.3 Device-iot服务 16](#_Toc73441504)

[4.3.1 功能描述 16](#_Toc73441505)

[4.3.2 功能设计/流程图 16](#_Toc73441506)

[4.3.3 对外接口 23](#_Toc73441507)

[4.3.4 具体实现核心概述 23](#_Toc73441508)

[4.4 中间件模块Device-action服务 24](#_Toc73441509)

[4.4.1 功能描述 24](#_Toc73441510)

[4.4.2 功能设计/流程图 25](#_Toc73441511)

[4.4.3 对外接口 25](#_Toc73441512)

[4.4.4 具体实现核心概述 26](#_Toc73441513)

[4.5 Control服务 26](#_Toc73441514)

[4.5.1 功能描述 26](#_Toc73441515)

[4.5.2 功能设计/流程图 27](#_Toc73441516)

[4.5.3 对外接口 27](#_Toc73441517)

[4.5.4 具体实现核心概述 27](#_Toc73441518)

[4.6 schedule服务 28](#_Toc73441519)

[4.6.1 功能描述 28](#_Toc73441520)

[4.6.2 功能设计/流程图 28](#_Toc73441521)

[4.6.3 对外接口 28](#_Toc73441522)

[4.6.4 具体实现核心概述 29](#_Toc73441523)

[4.7 iot core服务 29](#_Toc73441524)

[4.7.1 功能描述 29](#_Toc73441525)

[4.7.2 功能设计/流程图 30](#_Toc73441526)

[4.7.3 对外接口 30](#_Toc73441527)

[4.7.4 具体实现核心概述 30](#_Toc73441528)

[4.8 安全服务 30](#_Toc73441529)

[4.8.1 功能描述 30](#_Toc73441530)

[4.8.2 功能设计/流程图 31](#_Toc73441531)

[4.8.3 对外接口 31](#_Toc73441532)

[4.8.4 具体实现核心概述 32](#_Toc73441533)

[4.9 权限服务 32](#_Toc73441534)

[4.9.1 功能描述 32](#_Toc73441535)

[4.9.2 功能设计/流程图 32](#_Toc73441536)

[4.9.3 对外接口 34](#_Toc73441537)

[4.9.4 具体实现核心概述 34](#_Toc73441538)

# 云平台概要设计说明书

## 引言

### 编写背景

随着当今技术变革的日新月异，物联网核心器件需求高速增长。物联网作为通信行业新兴应用，在万物互联的大趋势下，市场规模将进一步扩大，人们对家居智能化的依赖需求越来越高，物联网是互联网行业发展的必然趋势。

基于以上理论，猿人从16年创建就一直在物联网的道路上不断摸索创新，智能插座SP20就是猿人最新开发的扩展项目。

### 编写目的

为强化公司内人员对智能插座云端服务的理解认知，需要对云端项目进行详细梳理。

本文详细阐述了云端服务中API接口、架构、数据库等设计及环境部署说明，作为开发人员及维护人员的指导性文档。

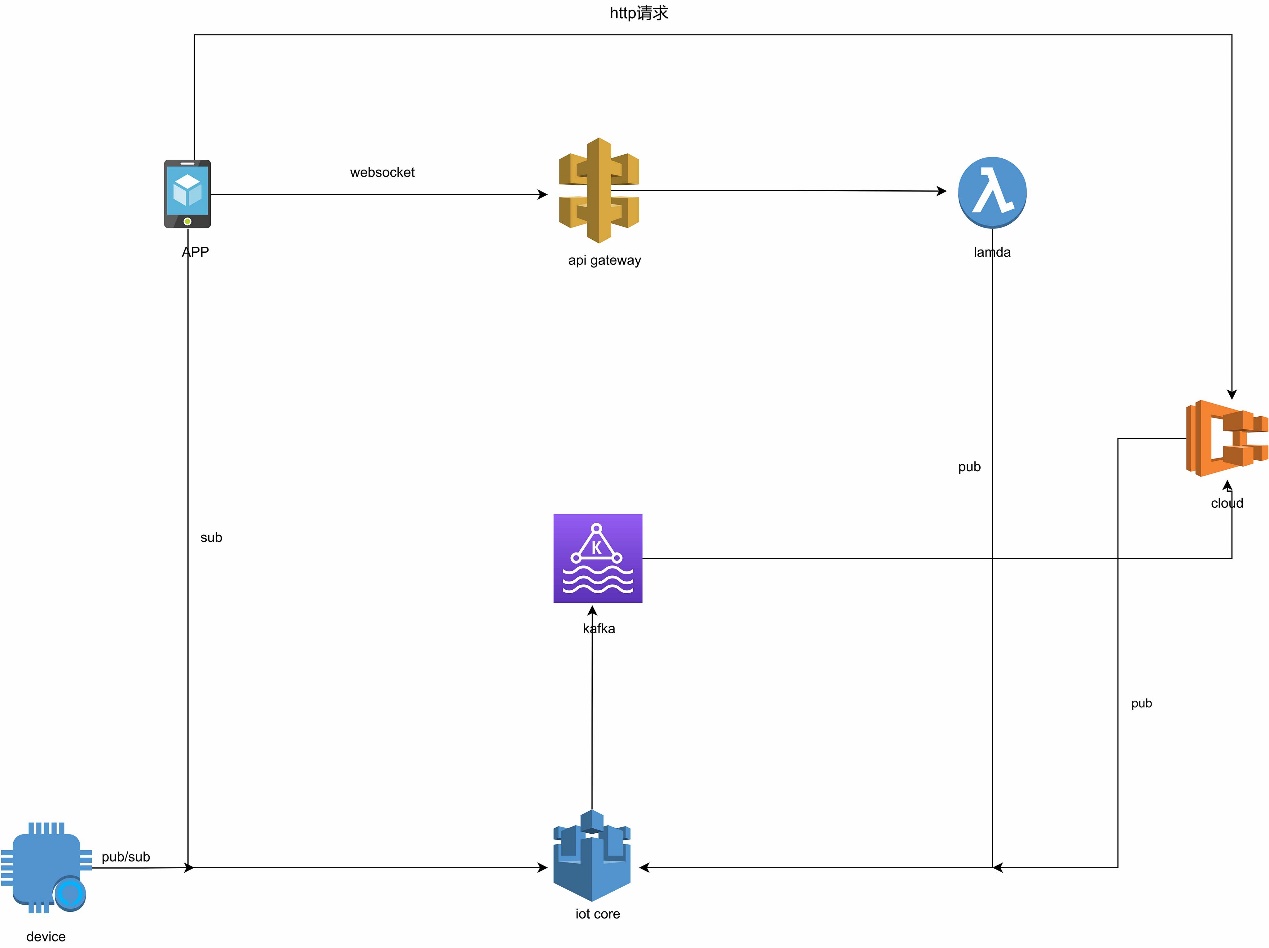
### 定义

### 参考规范

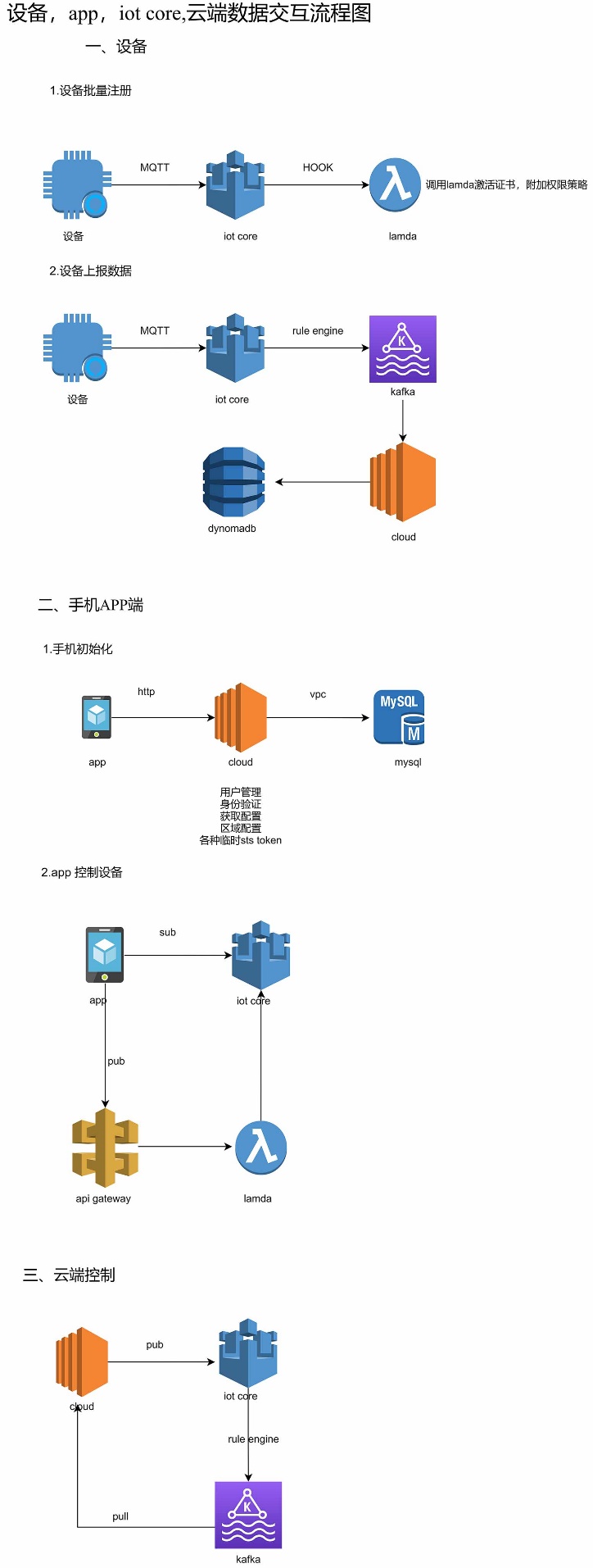
## 系统架构设计

### 架构设计

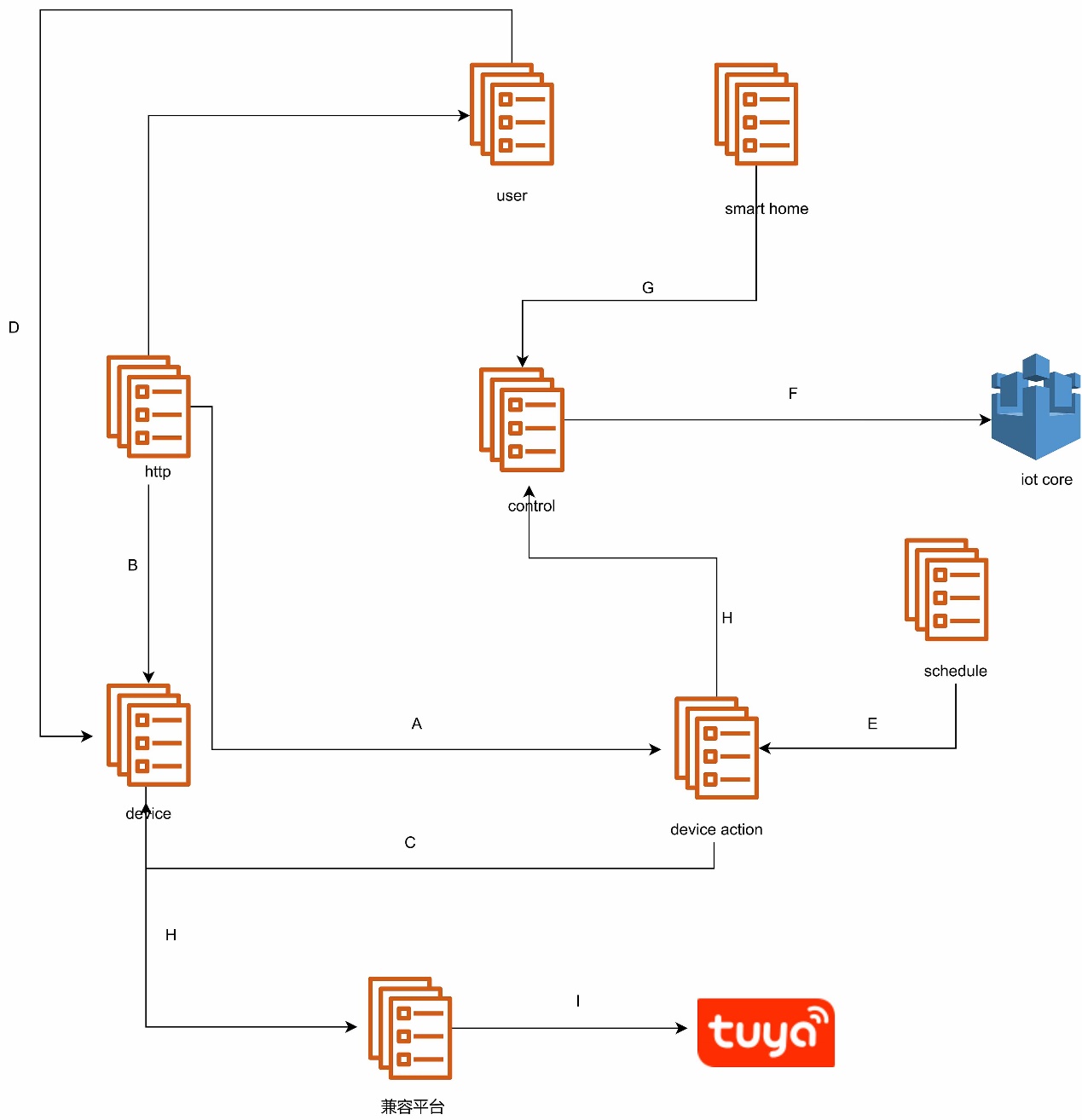
#### IOT-CLOUD



#### 数据交互



#### 微服务



### 关键技术/设计难点/解决方案

关键技术：使用mqtt作为交互中间件，这是一种基于发布/订阅的轻量级通讯协议。

设计难点：高并发及高可用一直是软件开发功能上的一道难关，未来的SP20插座可能会突破千万级的设备量，qps会达到百万级，这是设计中最大的难点。

解决方案：智能家居在中国乃至世界也是未来的开发主流，市面上常用的mqtt服务有两个，emqx和aws iot，经多方调研及验证，相比下aws iot服务链路全，安全性高，维护速度快，基于此考虑最终选择使用aws iot作为开发的核心服务。

### 开发环境配置

服务器版本：centos8、windows10

开发语言版本：golang 1.16

所需其他安装：

java 1.8

kafka 2.6.1

Mysql 5.7

Redis 6.2.1

aws-cli v2

### 测试环境配置

服务器版本：centos8

开发语言版本：golang 1.16

所需其他安装：

java 1.8

kafka 2.6.1

Mysql 5.7

Redis 6.2.1

aws-cli v2

### 生产环境配置

服务器版本：centos8

开发语言版本：golang 1.16

所需其他安装：

java 1.8

kafka 2.6.1

Mysql 5.7

Redis 6.2.1

aws-cli v2

## 数据库设计

### 数据库环境说明

依赖环境：Linux、Windows

数据库使用：Mysql（5.7），Redis（6.2.1）

依赖插件：Navicate（Mysql）

### 数据库命名规则

1. 使用小写英文字母
2. 使用下划线做连接符
3. 表名要通俗易懂

### 表汇总

1、device 设备信息表

2、user\_device 用户绑定的设备表

3、user 用户基本信息表

4、user\_info 用户扩展信息表

5、user\_automation 用户自定义情景模式

6、device\_automation 设备情景模式操作

### 安全性设计

#### 防止用户直接操作数据库的方法

为避免用户直接操作数据库，做了如下预防方案：

1. 使用https加密传输，传输数据无法通过抓包获取。
2. 在接口业务层中对sql注入做中间件预防，避免用户在输入框使用sql注入攻击。

#### 用户账号密码的加密方法

1、MD5加密

#### 数据库管理与维护说明

1. 设置管理权限
2. 监控数据库的状态，出现问题及时报警并通知相关人员
3. 做好数据备份

## 功能设计

### 功能汇总

1. http服务
2. Device-iot服务
3. Device action服务
4. Control服务
5. Schedule服务
6. IoT-core服务
7. 安全服务
8. 权限服务

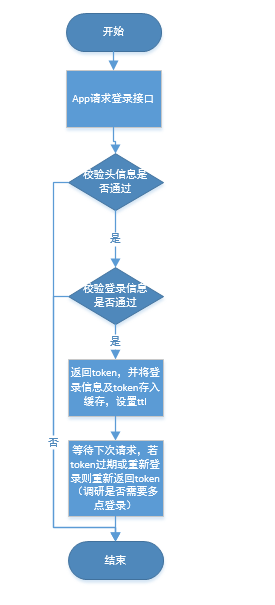
### http服务

#### 功能描述

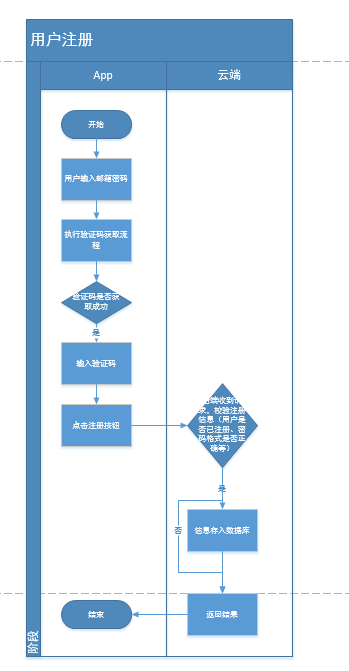
http服务提供给app的接口，包括登录验证，加密，功能接口等，此外，还有App接入时重定向到正确的区域，Sts token管理等功能。

#### 功能设计/流程图

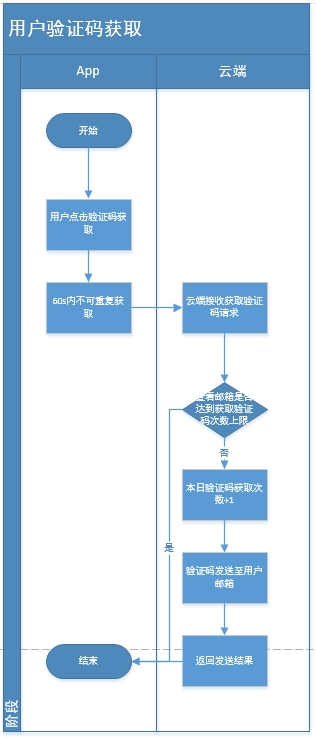
1. 用户登录



1. 用户注册



1. 用户验证码获取



#### 对外接口

|  |  |  |  |
| --- | --- | --- | --- |
|  | **功能** | **协议** | **描述** |
| 1 | 用户注册 | http |  |
| 2 | 用户登录 | http |  |
| 3 | 用户注销 | http |  |
| 4 | 用户登出 | http |  |
| 5 | 用户验证码获取 | http |  |

#### 具体实现核心概述

经多方调研，框架选择使用go-kit搭建。go-kit是用golang语言开发的一款轻量级的框架，严格意义上来说，go-kit其实是一个golang的工具集，开发者可以更具自己的需求，自由使用里面的工具，具有高性能，使用快捷方便等优点。

在http服务使用这个框架，可以做到模块间最大程度的解耦，使得模块与模块之间互不影响，对之后代码的维护与扩展打好基础。

### Device-iot服务

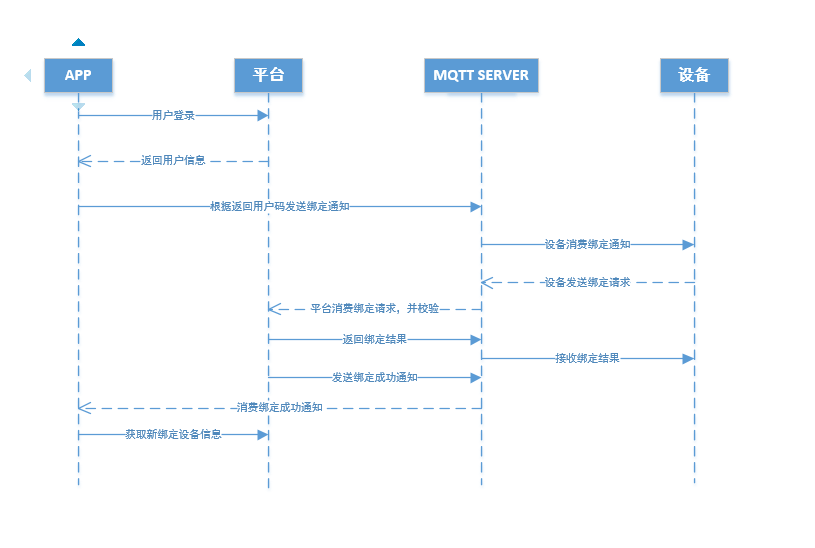
#### 功能描述

Device-iot服务是对于设备相关的一些管理。

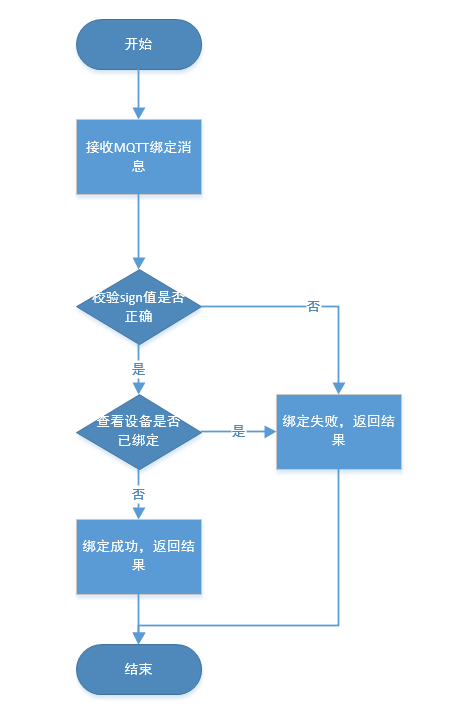
主要功能包括入网，删除，查询，更新，iot证书策略附加、权限验证、OTA升级、设备消息，一些状态变化，遥测数据持久化消息等。

#### 功能设计/流程图

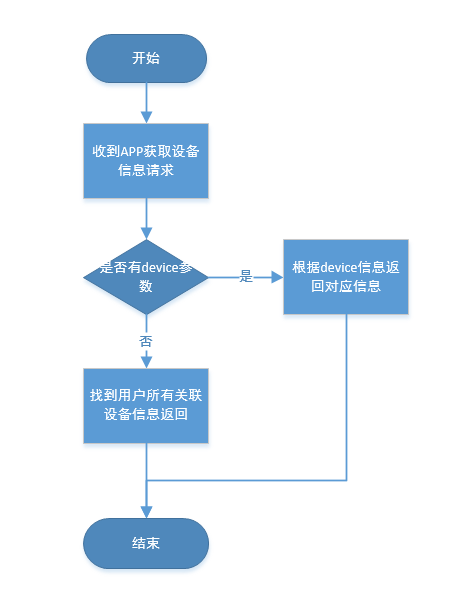
1. 设备绑定流程



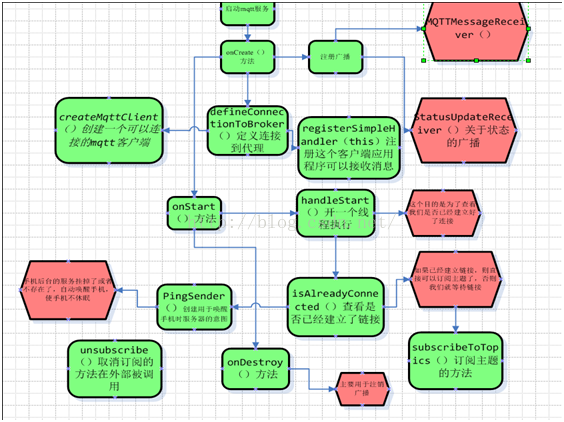
1. 设备绑定校验



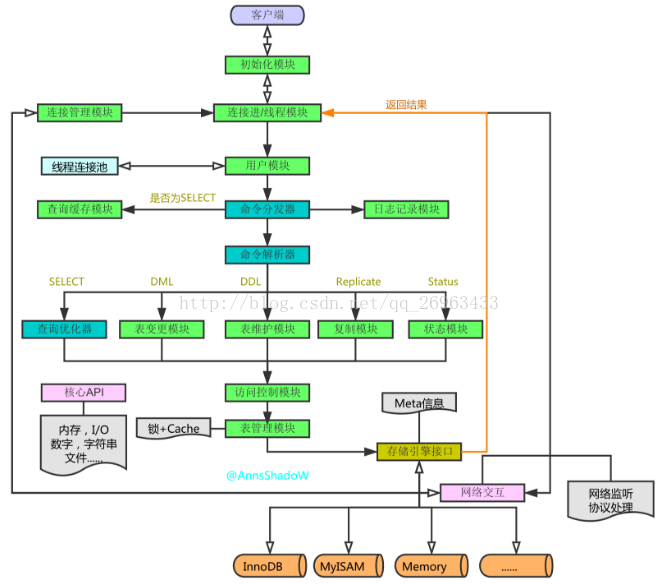
1. 用户获取设备列表



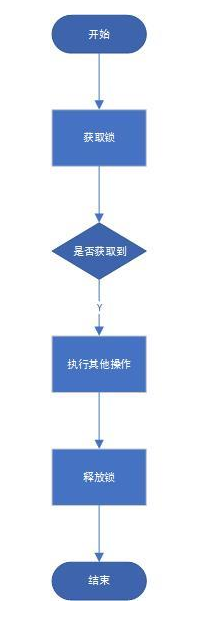
4、mqtt流程图

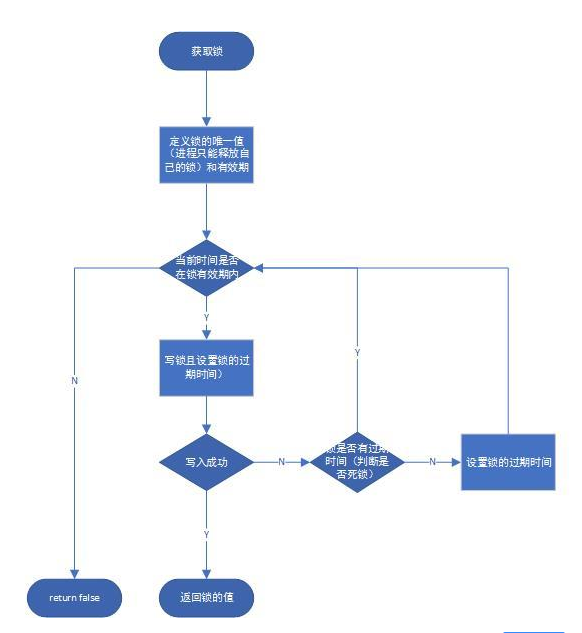


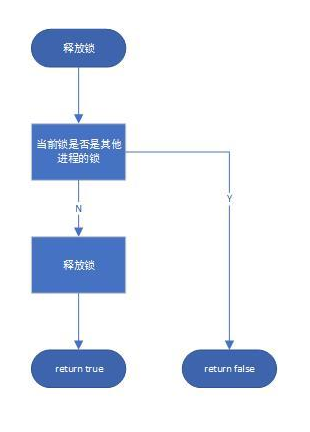
5、mysql流程图



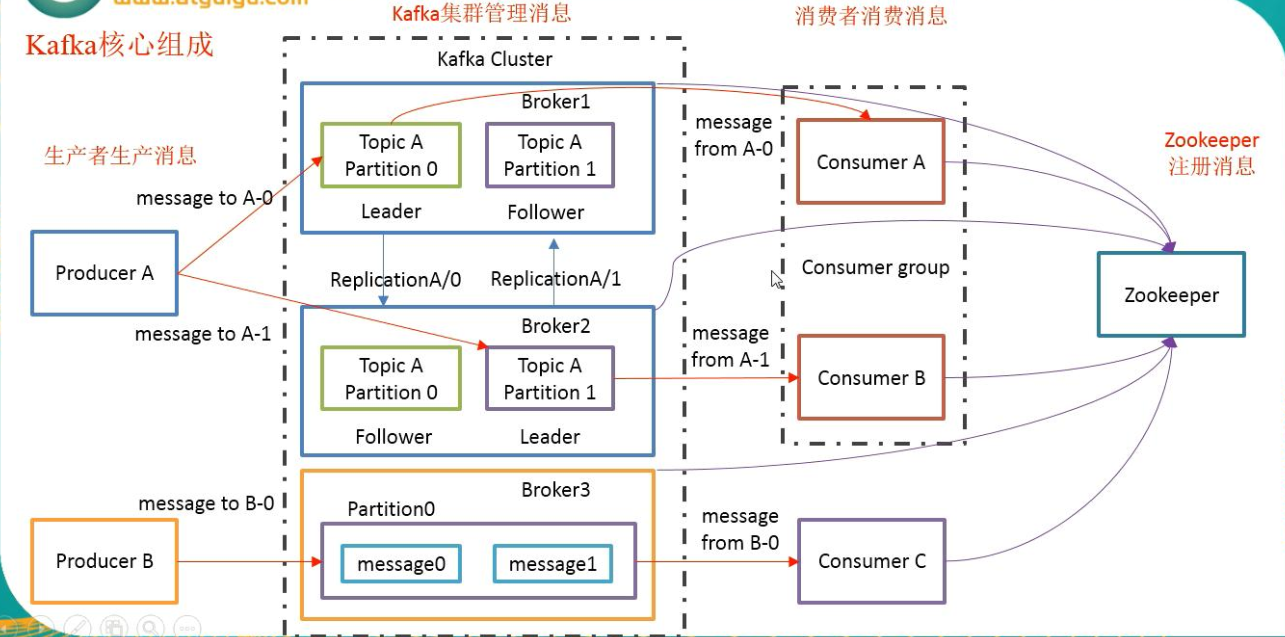
6、redis流程图（主要是锁）







7、kafka流程图



#### 对外接口

|  |  |  |  |
| --- | --- | --- | --- |
|  | **功能** | **协议** | **描述** |
| 1 | 用户获取设备信息 | grpc | 用户可根据参数获取全部绑定列表或单个信息 |
| 2 | 设备绑定 | grpc |  |
| 3 | 设备上报心跳 | mqtt |  |
| 4 | 设备上报状态 | mqtt |  |
| 5 | 控制设备开关 | grpc |  |
| 6 | ota升级 | grpc |  |
| 7 | iot证书策略附加 | grpc |  |

#### 具体实现核心概述

这个部分一共可以分为三个部分，设备的增删改查部分，iot证书策略以及权限部分，以及设备状态部分。

设备部分：该部分主要是设备的增删改查的操作，涉及到设备的所有控制相关的逻辑，包括入网、控制、解绑、更新以及查询等，其他所有地方对于设备的控制，都是通过这部分的grpc接口来对外提供，做到控制的统一管理，便于维护。

Iot证书策略以及权限部分：主要是针对AWS平台IOT证书的权限的管理，包括对证书的验证，生成，删除以及修改等，还有对于证书策略的验证，增加和修改，其实现都是与AWS的IAM服务以及策略服务对接。

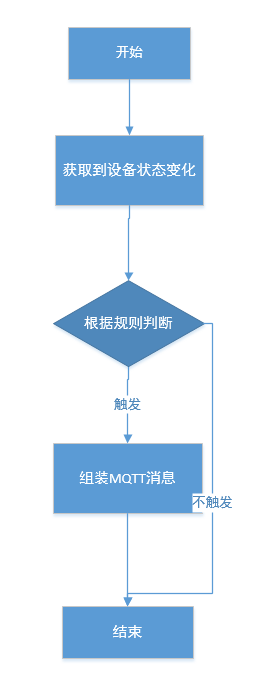
设备状态部分：这部分主要是对于设备状态的维护与处理。其主要作用是对于设备状态数据的持久化，包括缓存和数据库两方面，都需要进行维护。其他服务监控到设备状态发生了变化时，会将消息推送到本服务监听的kafka的队列里，这边从kafka的队列里取到消息后，解析消息，然后把缓存以及数据库里对应设备的状态进行更新，当其他服务来本服务获取设备相关数据时，我们会先从缓存中获取数据，如果缓存中没有，则会从数据库里获取，最后将数据库存在缓存里，做好对数据库的降压。

### 中间件模块Device-action服务

#### 功能描述

提供以设备为基础的定制化服务，如情景模式，automation, 通知，报警等，提供情景模式新建、更新和查询，执行定时情景模式的触发逻辑，读取设备功能状态消息，转发给app，读取异步状态消息，转发给automation，触发基于设备的通知和报警，接收set、get、ack、push等消息，写一些触发逻辑和日志记录。

#### 功能设计/流程图



#### 对外接口

|  |  |  |  |
| --- | --- | --- | --- |
|  | **功能** | **协议** | **描述** |
| 1 | 新增设备规则 | grpc |  |
| 2 | 删除设备规则 | grpc |  |
| 3 | 修改设备规则 | grpc |  |
| 4 | 查询设备规则 | grpc |  |

#### 具体实现核心概述

该服务主要是实现定制化的功能，例如定时控制设备，根据外部条件（比如天气，温度，湿度，设备电量等）控制设备，还有设备之间的互相联动，例如当卧室的灯打开了，就把空调也打开这类似的场景功能。我们会对外提供接口，让用户可以自己设置规则（根据我们提供的一些条件），我们保存这些规则，当系统收到设备状态的变更消息时，就根据一定的规则去判断设备当前的状态需不需要触发一定的逻辑，需要的话就调用device服务去发送控制命令，实现自定义的功能。

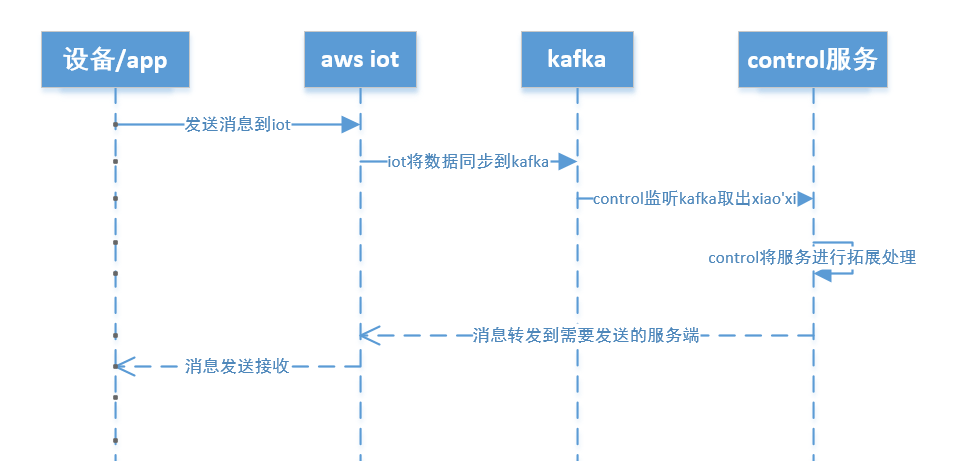
除此之外，该服务还有记录日志和报警的功能，收到设备的消息时，会记录对应的日志，对于异常消息，会触发报警，比如发邮件或者短信之类的通知功能（其他服务提供）。

### Control服务

#### 功能描述

用于处理iot core海量命令，并进行处理转发。

#### 功能设计/流程图



#### 对外接口

无

#### 具体实现核心概述

1. 处理设备通过iot core发送给云端的专有处理命令

2、 处理app通过iot core发送给云端的专有处理命令

3、 读取KafKa并处理消息，然后根据业务放入kafka,提供下游服务使用的消息

4、 处理设备的一些配置信息，调用device服务持久化

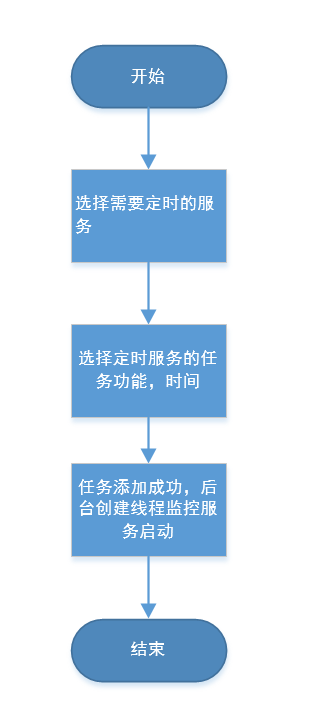
5、 处理设备上下线时间，通知app设备上下线事件

### schedule服务

#### 功能描述

提供周期性事务执行服务，将其余服务需要定时工作的任务按配置时间运行。配置人员可以精确工作时间到日，时：分：秒。不同服务可配置的功能也不同。

#### 功能设计/流程图



#### 对外接口

|  |  |  |  |
| --- | --- | --- | --- |
|  | **功能** | **协议** | **描述** |
| 1 | 查询当前任务 | http |  |
| 2 | 任务添加 | http |  |
| 3 | 任务删除 | http |  |
| 4 | 任务信息修改 | http |  |
| 5 | 控制设备开关 | http |  |

#### 具体实现核心概述

核心实现分为两部分，对外接口控制与内部任务调度。

对外接口：采用动静分离，对前端暴露接口，动态调度任务信息，前端页面可实现任务的增删改查，各服务通过接口方法上报信息，将服务列表缓存到云端。

内部任务调度：任务增加后，系统会自动分配协程调度任务，任务信息会存入到缓存中，设置ttl为任务截止时间，同时设置定时器，定时结束启动任务，当服务重启后，会从缓存拉取任务信息，重新进行定时任务调度，避免任务执行失败。

### iot core服务

#### 功能描述

mqtt是一款基于发布/订阅的轻量级通讯系统，iot core提供了高效稳定的mqtt服务连接，通过构筑的连接池，分配回收连接对象，在减少cpu消耗的前提下保证服务正常运转

#### 功能设计/流程图

无

#### 对外接口

无

#### 具体实现核心概述

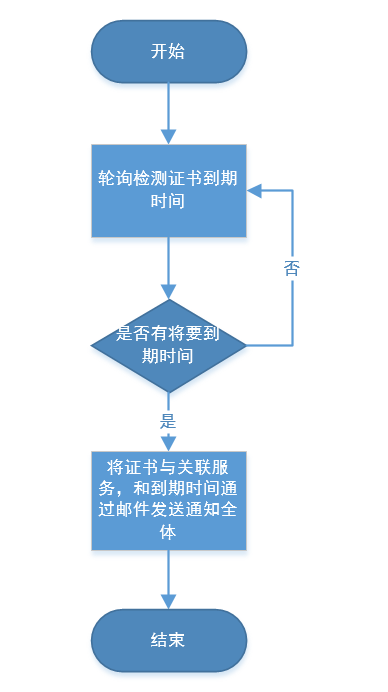
iot core服务主要运用了协程池原理，在服务启动时，启动配置目标的mqtt服务连接数，保持空闲等待，在有客户端连接进入的时候，获取空闲的连接池，分配连接，并将连接配置为使用中，当连接关闭时，将连接放入连接池，连接对象状态改为闲置，全程使用互斥锁释放分配。

### 安全服务

#### 功能描述

维护各服务的证书管理。

#### 功能设计/流程图



#### 对外接口

|  |  |  |  |
| --- | --- | --- | --- |
|  | **功能** | **协议** | **描述** |
| 1 | 查询证书列表 | http |  |
| 2 | 证书添加 | http |  |
| 3 | 证书状态修改 | http |  |

#### 具体实现核心概述

维护各服务所使用证书，将服务与证书的调用关系关联，配置证书到期时间校验，如证书到期两月或一月前提前发出邮件通知，避免检查不当导致证书到期，服务无法使用的情况。

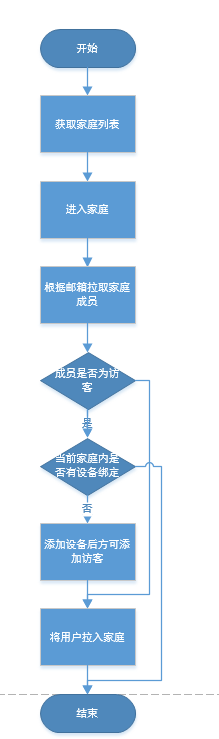
### 权限服务

#### 功能描述

管理用户、设备、家庭组的关联权限，以及家庭组的配置。

#### 功能设计/流程图

家庭拉人：



#### 对外接口

|  |  |  |  |
| --- | --- | --- | --- |
|  | **功能** | **协议** | **描述** |
| 1 | 家庭组设备列表获取 | http |  |
| 2 | 家庭组人员列表获取 | http |  |
| 3 | 家庭组添加成员 | http |  |
| 4 | 家庭组删除成员 | http |  |
| 5 | 家庭组添加设备 | http |  |
| 6 | 家庭组删除设备 | http |  |

#### 具体实现核心概述

权限两类，设备权限、家庭成员分类。

家庭成员分为超级管理员、普通管理员、访客。

超级管理员可添加删除普通管理员和访客及家庭关联设备，普通管理员可添加删除访客及家庭关联设备，家庭关系拉取全部依赖于用户邮箱。

访客只有设备开关权限，无权调用设备场景化功能；访客无权增删家庭内设备。