对外网关服务

> 本章内容描述，第三方系统访问内部服务时，需要的配置

概述

> 外部系统访问本系统的api，可以采用两种认证方法。
>
> 1、简单认证。通过本系统颁发的token，在访问本系统api时，header中添加有效的token访问。
>
> 2、加密认证。通过本系统提供的加密api，对request中的数据加密，并在header中提供加密认证的appKey访问。

认证规则

1.服务调用方

1.申请APP，获取appKey/appToken

描述：通过**APP管理**功能注册APP，给服务调用方返回

appKey（24位字符串数字、大小写字母组合）/appToken（32位字符串数字、大小写字母组合）（用于对称签名校验）

说明：简单认证只生成appToken，签名认证会生成appKey和appToken。

2.调用ACCA提供的SDK，访问目标服务

描述：

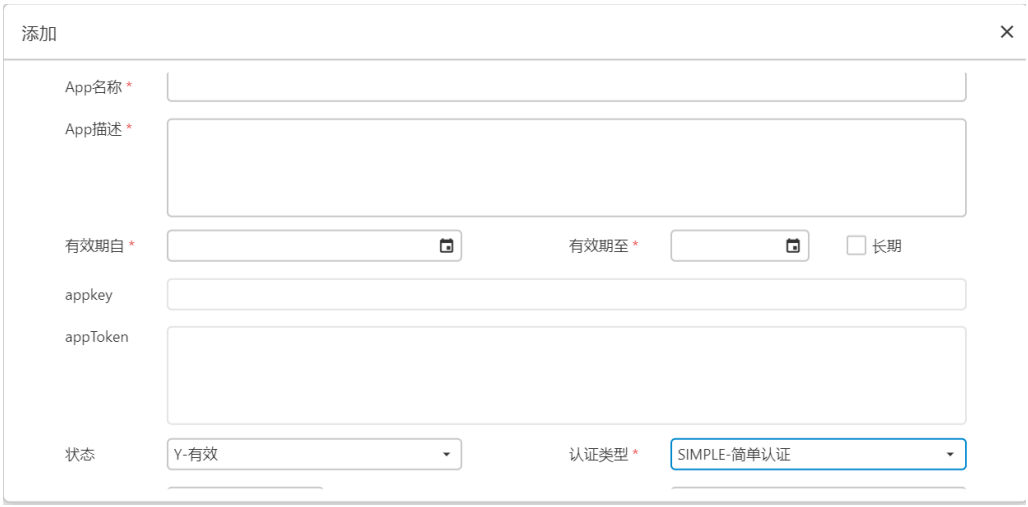
服务调用方，通过封装appKey及其它请求参数，用appToken进行HmacSHA256签名，访问目标服务

2.服务提供方-前端页面

维护数据

1、基础维护。维护app名称，app描述，有效期，流量控制次数，认证类型等信息，自动生成appToken。

2、分配功能。维护url权限。



功能列表

1.APP管理【必选】

功能描述：第三方申请APP，获取appKey/appToken，提供新增、删除功能

关键字：**APP名称/appkey/appToken**/app描述

2.分组管理【必选】

功能描述：创建 API 需要先创建分组，方便管理 API ，提供新增、API管理、删除功能

关键字：**分组名称/**分组描述

3.API管理【必选】

功能描述：管理需要对外暴露的API，提供新增、上线、下线、授权、删除功能

关键字：**API名称/apiPath/httpMethod/**api描述

4.IP控制【可选】

功能描述：黑白名单控制

5.流量控制【可选】

功能描述：第三方请求流量控制

6.SDK生成【可选】

功能描述：已发布的API自动导出SDK

3.服务提供方-后端

后台处理逻辑如下：

网关收到【**对外域名**】的请求后

1.判断该appKey或appToken的访问url是否存在

2.IP控制判断【可选】

3.流量控制判断【可选】

4. 根据请求参数对请求进行签名校验或者appToken简单校验

5. Token校验【可选】

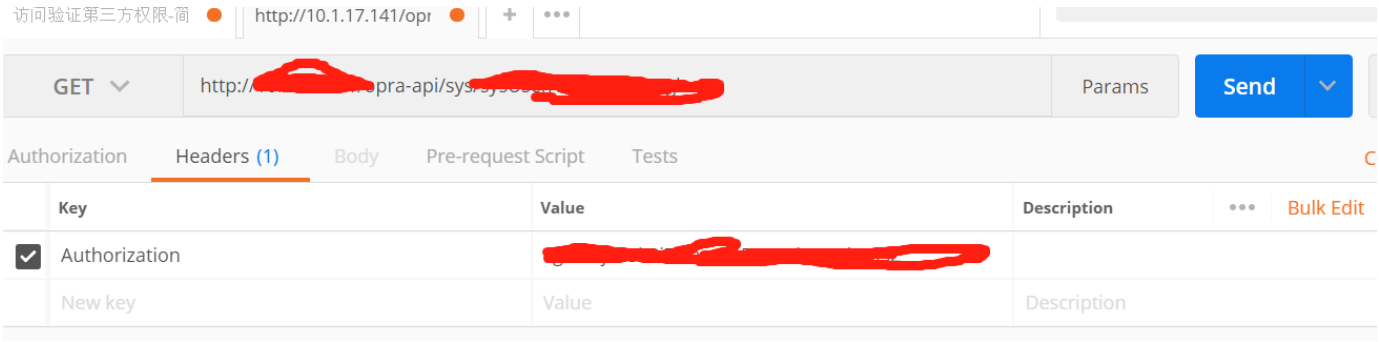
6. 校验成功则正常访问后端请求

4. 请求方式

4.1AppToken简单认证访问

通过在请求header中增加：

Authorization: {appToken}



4.2AppKey及AppToken签名认证访问

4.2.1客户端生成签名

通过在请求header中增加：

user-agent: ACCA-API

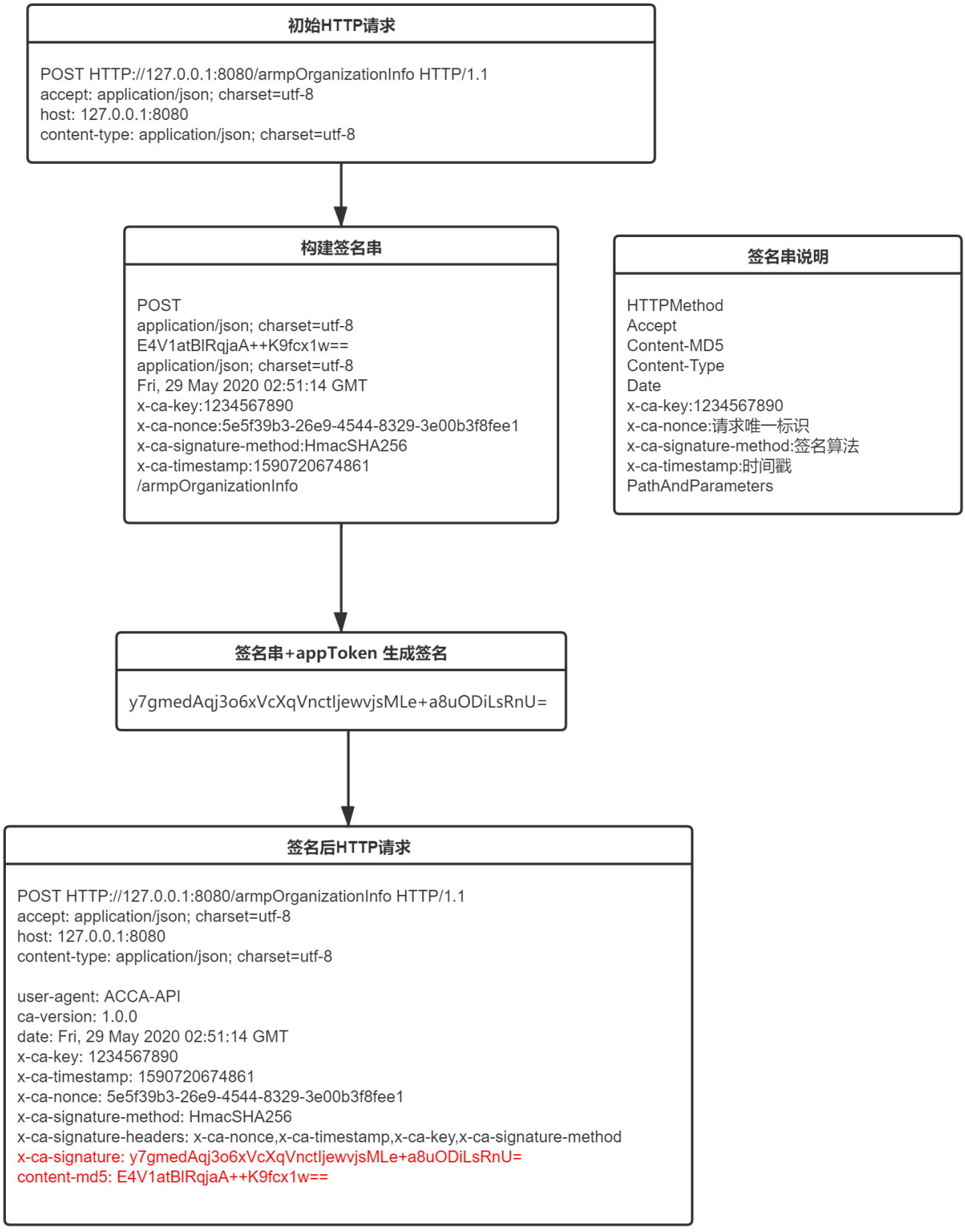
x-ca-key: {appKey}

x-ca-signature: {signature}

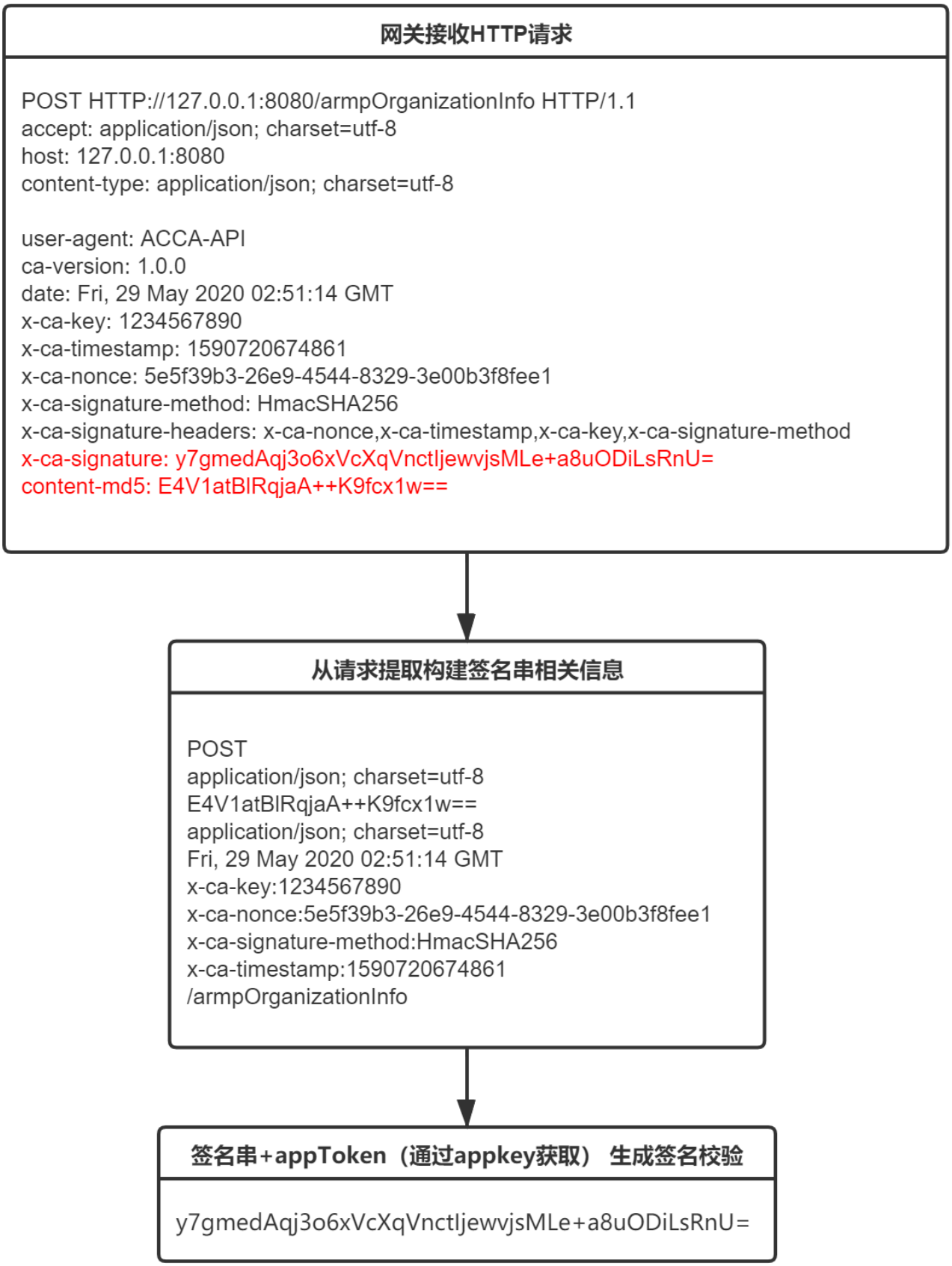
content-md5: {content-md5}

其中user-agent: ACCA-API为签名认证标识，appKey为创建自动生成的appKey，signature为请求签名串，content-md5为SignUtil.base64AndMD5签名。

4.2.2客户端生成签名


初始HTTP请求
POST HTTP://127.0.0.1:8080/armpOrganizationInfo HTTP/1.1
accept: application/json; charset=utf-8
host: 127.0.0.1:8080
content-type: application/json; charset=utf-8
构建签名串
POST
application/json; charset=utf-8
E4V1atBlRqjaA++K9fcx1w==
application/json; charset=utf-8
Fri, 29 May 2020 02:51:14 GMT
x-ca-key:1234567890
x-ca-nonce:5e5f39b3-26e9-4544-8329-3e00b3f8fee1
x-ca-signature-method:HmacSHA256
x-ca-timestamp:1590720674861
/armpOrganizationInfo
签名串说明
HTTPMethod
Accept
Content-MD5
Content-Type
Date
x-ca-key:1234567890
x-ca-nonce:请求唯一标识
x-ca-signature-method:签名算法
x-ca-timestamp:时间戳
PathAndParameters
签名串+appToken 生成签名
y7gmedAqj3o6xVcXqVnctIjewvjsMLe+a8uODiLsRnU=
签名后HTTP请求
POST HTTP://127.0.0.1:8080/armpOrganizationInfo HTTP/1.1
accept: application/json; charset=utf-8
host: 127.0.0.1:8080
content-type: application/json; charset=utf-8
user-agent: ACCA-API
ca-version: 1.0.0
date: Fri, 29 May 2020 02:51:14 GMT
x-ca-key: 1234567890
x-ca-timestamp: 1590720674861
x-ca-nonce: 5e5f39b3-26e9-4544-8329-3e00b3f8fee1
x-ca-signature-method: HmacSHA256
x-ca-signature-headers: x-ca-nonce,x-ca-timestamp,x-ca-key,x-ca-signature-method
x-ca-signature: y7gmedAqj3o6xVcXqVnctIjewvjsMLe+a8uODiLsRnU=
content-md5: E4V1atBlRqjaA++K9fcx1w==

4.2.3网关校验签名


网关接收HTTP请求
POST HTTP://127.0.0.1:8080/armpOrganizationInfo HTTP/1.1
accept: application/json; charset=utf-8
host: 127.0.0.1:8080
content-type: application/json; charset=utf-8
user-agent: ACCA-API
ca-version: 1.0.0
date: Fri, 29 May 2020 02:51:14 GMT
x-ca-key: 1234567890
x-ca-timestamp: 1590720674861
x-ca-nonce: 5e5f39b3-26e9-4544-8329-3e00b3f8fee1
x-ca-signature-method: HmacSHA256
x-ca-signature-headers: x-ca-nonce,x-ca-timestamp,x-ca-key,x-ca-signature-method
x-ca-signature: y7gmedAqj3o6xVcXqVnctIjewvjsMLe+a8uODiLsRnU=
content-md5: E4V1atBlRqjaA++K9fcx1w==
从请求提取构建签名串相关信息
POST
application/json; charset=utf-8
E4V1atBlRqjaA++K9fcx1w==
application/json; charset=utf-8
Fri, 29 May 2020 02:51:14 GMT
x-ca-key:1234567890
x-ca-nonce:5e5f39b3-26e9-4544-8329-3e00b3f8fee1
x-ca-signature-method:HmacSHA256
x-ca-timestamp:1590720674861
/armpOrganizationInfo
签名串+appToken（通过appkey获取） 生成签名校验
y7gmedAqj3o6xVcXqVnctIjewvjsMLe+a8uODiLsRnU=