应用托管多租户SaaS对接方案

目录

变更记录

版本号	变更点	变更人	变更时间
v0.9	1. 新增免密登录接口的安全说明 2. 新增获取手机号接口的安全说明	林示	2018.11.16
v0.8	1. 免密登录接口：租户组织架构中员工的唯一ID，tenantSubId修改为tenantSubUserId； 2. 所有接口：添加appId字段，该字段为IoT平台标识一个租户一次购买应用的唯一ID； 3. 生产租户接口：添加appType字段； 4. 所有接口：添加id用于幂等校验； 5. 新增获取用户手机号API及调用方式；	林示	2018.11.06

方案概述

名词解释

【AppKey/AppSecret】

AppKey和AppSecret是成对出现的账号，是标识应用的唯一身份信息，也是作为接口调用的加签延签密钥

【多租户SaaS】

是一种软件单个实例部署，但是向多个租户提供服务的框架，一般这类SaaS都以开立账号的方式添加租户。

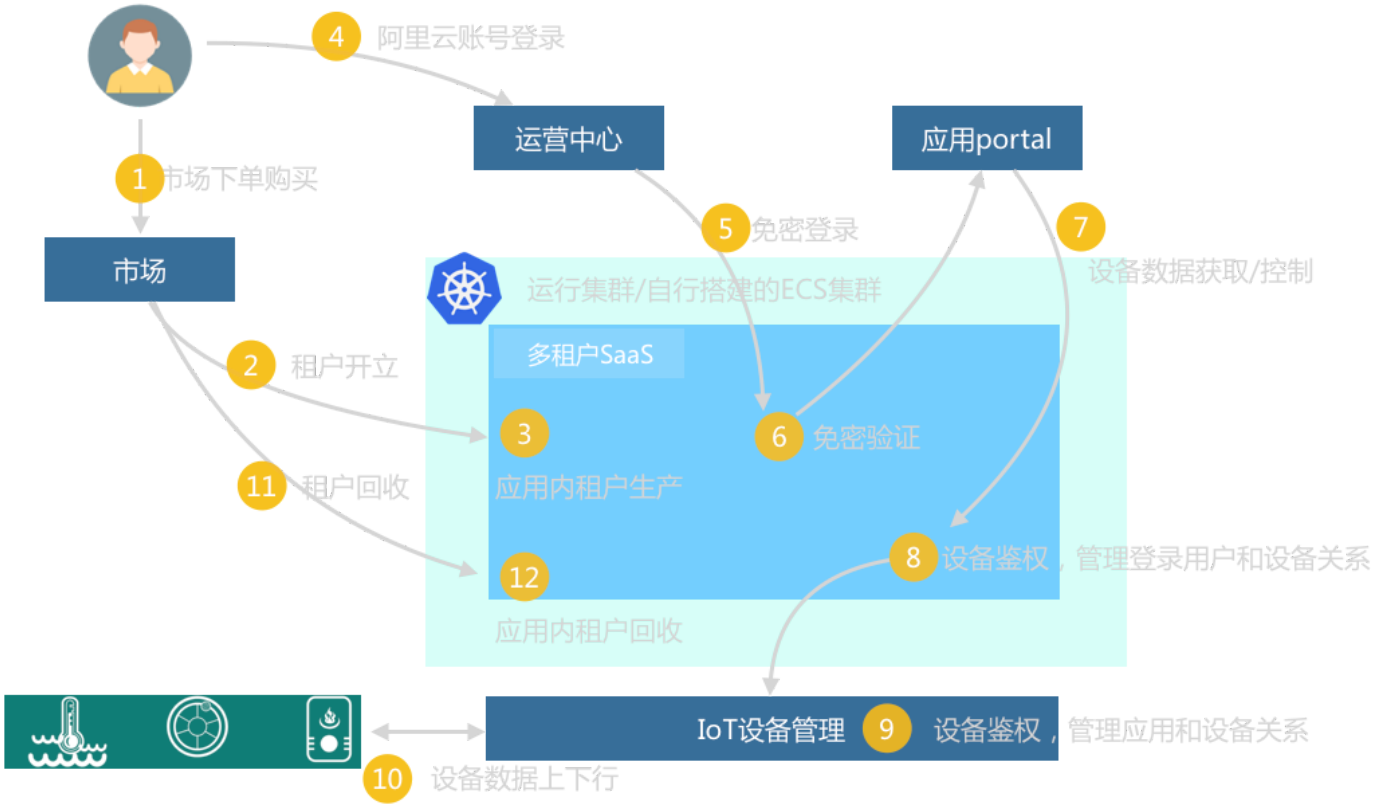
【k8s运行集群】

核心的特点就是能够自主的管理容器来保证云平台中的容器按照用户的期望状态运行着（比如用户想让apache一直运行，用户不需要关心怎么去做，Kubernetes会自动去监控，然后去重启，总之，让apache一直提供服务），您可以通过应用托管的部署工具进行部署，详情参见【应用托管】

【自行搭建的ECS集群】

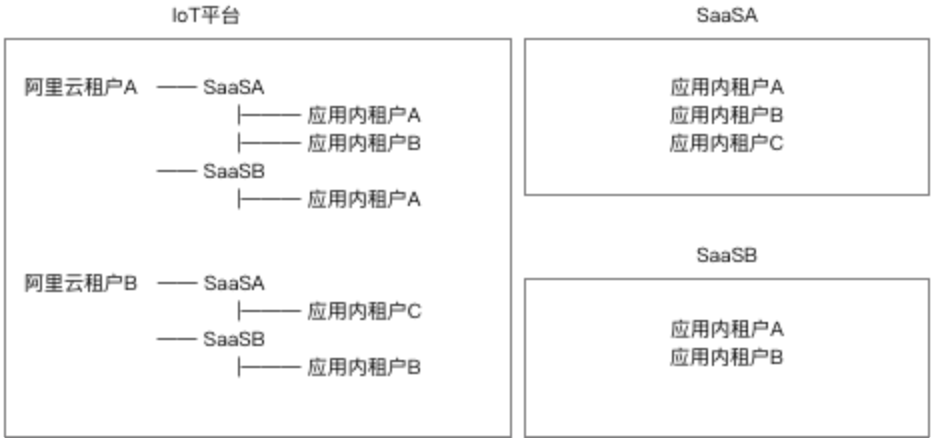
指用户自己通过阿里云ECS、RDS等云产品搭建的SaaS服务集群，该集群完全依赖用户自行管理及维护；

整体链路


4 阿里云账号登录
运营中心
应用portal
1 市场下单购买
5 免密登录
7 设备数据获取/控制
市场
运行集群/自行搭建的ECS集群
多租户SaaS
2 租户开立
3
6 免密验证
11 租户回收
应用内租户生产
8 设备鉴权，管理登录用户和设备关系
12
应用内租户回收
IoT设备管理
9 设备鉴权，管理应用和设备关系
10 设备数据上下行

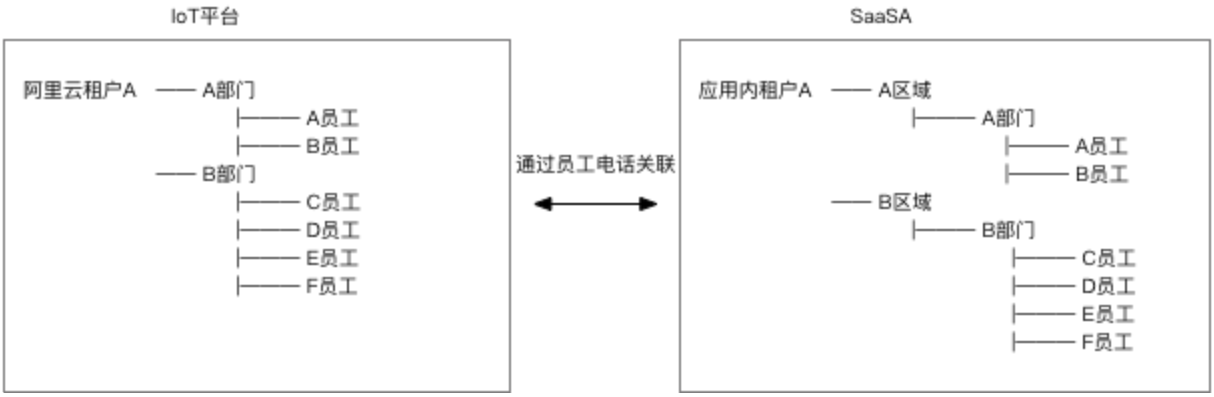
功能边界

租户管理



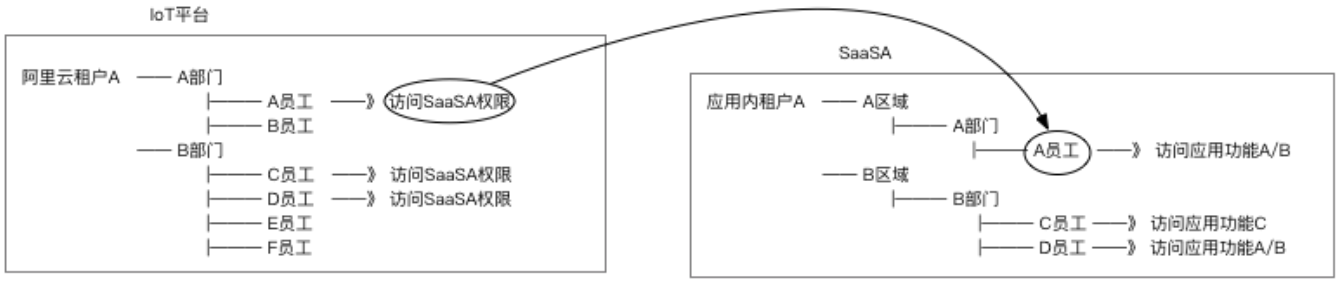
IoT平台管理阿里云租户与应用内租户的映射关系（支持一个阿里云租户拥有同一个应用的多个租户），SaaS管理应用内部的租户结构

组织架构



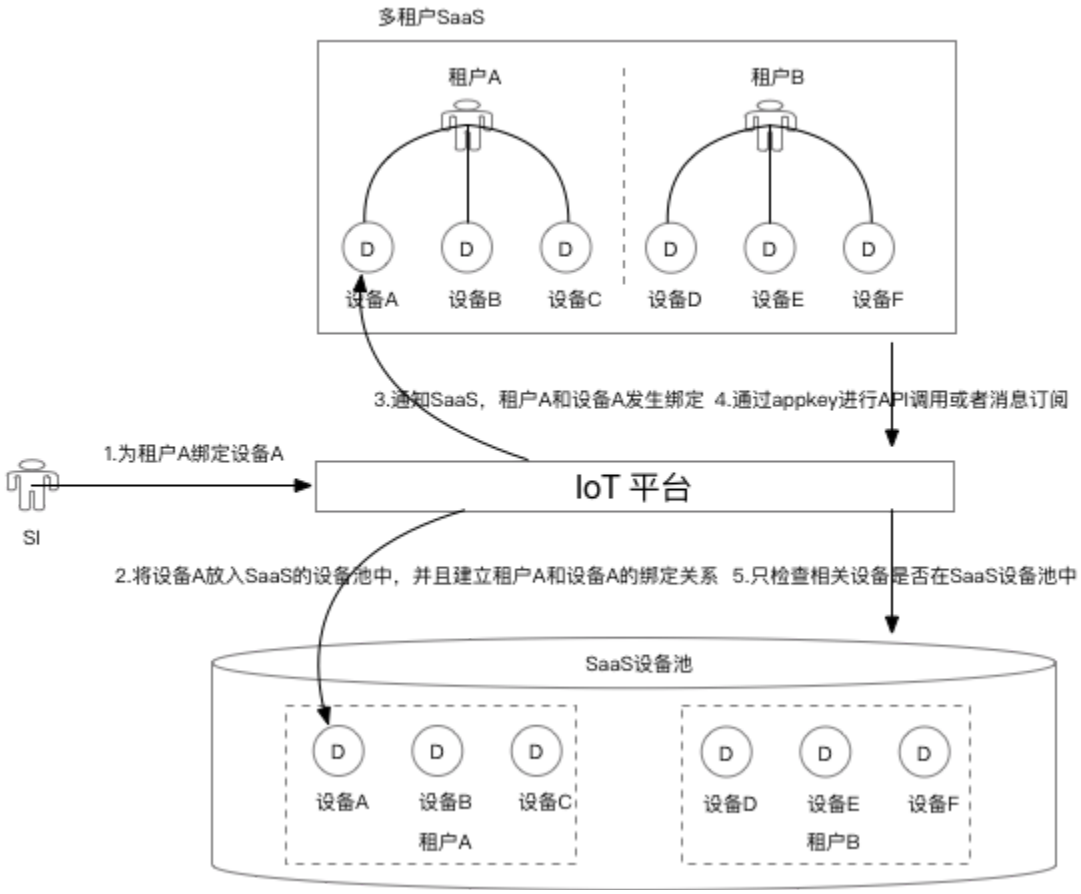
本期无需对接IoT平台的组织架构体系，SaaS内部自行管理组织架构，但是两边的组织架构需要能够定位到同一个员工身份， 目前通过员工电话作为识别的唯一标识；

权限管理



IoT平台管理租户及其组织架构中的员工是否有权限访问应用，SaaS管理租户及组织架构中员工访问功能列表；

设备管理



- SaaS管理租户和设备之间的权限关系
- IoT平台管理SaaS和设备之间的权限关系

注意：

- 通知SaaS绑定/解绑设备请查看【对接流程.接口开发.绑定设备/解绑设备】
- 应用如何使用设备API或者订阅消息查看https://help.aliyun.com/document_detail/93223.html

Appkey申请

AppKey的来源

- 应用部署在应用托管平台中，系统会为部署的每个应用颁发AppKey，AppKey通过环境变量方式注入至业务代码运行的容器中，供代码获取并使用；

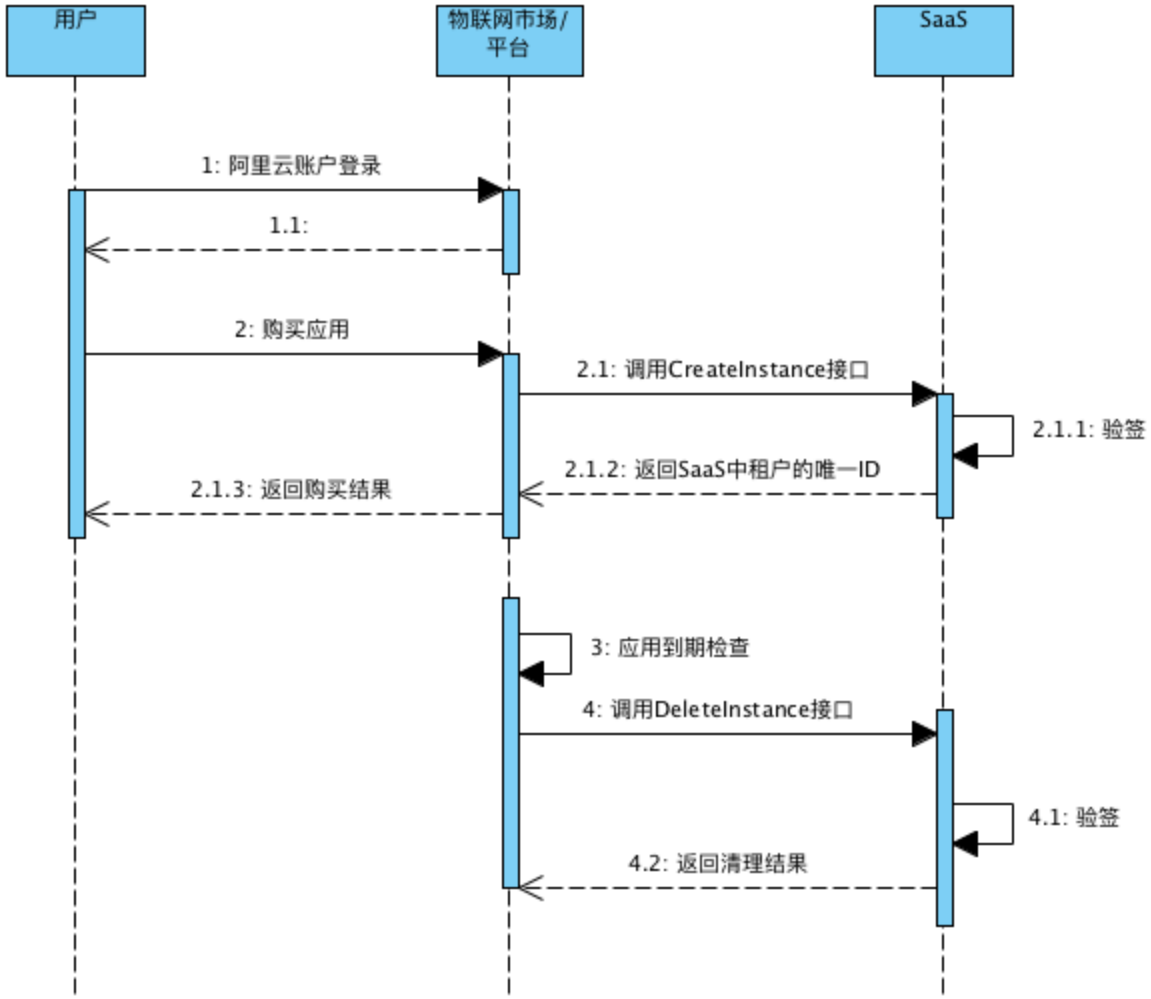
- 应用部署在自行搭建的ECS集群中，则需要人工申请，可在ISV技术对接群中@支持同学（申请时提供账户ID）；

生产租户

场景描述

用户（阿里云账户体系）在物联网市场下单购买应用后需要通知SaaS为当前用户开立一个可访问SaaS服务的租户（账户），用户可以通过点击物联网市场中已购买的应用或者直接访问SaaS提供的公网域名来访问应用，当用户购买的应用到期后系统会自动发起SaaS租户的回收操作。所以您需要实现**CreateInstance**接口，来实现用户购买应用场景；

调用时序



接口说明

CreateInstance生产服务

protocal	httpMethod
HTTPS	POST

请求参数

参数	类型	必填	描述
id	String	是	✓该次访问唯一标示符（幂等验证ID）
tenantId	String	是	IoT平台标识一个租户的唯一ID
appId	String	是	应用唯一ID， 一个租户可以重复购买一款软件， 每次购买appId都不同
appType	String	是	”TRIAL”:试用， 用户可以有一个短暂的试用期， 该功能在商品上架时ISV可以选择是否提供试用逻辑 ”BUY”：正式购买，用户正式下单购买

返回参数

参数	类型	描述
code	Integer	调用成功返回200； 若调用失败返回203；
message	String	code返回203时填写具体失败原因，code返回200填写success
userId	String	SaaS标识一个租户的唯一ID， 存在一个租户购买多次， 则SaaS返回多个userId

验签说明

参见【验签说明】

返回示例

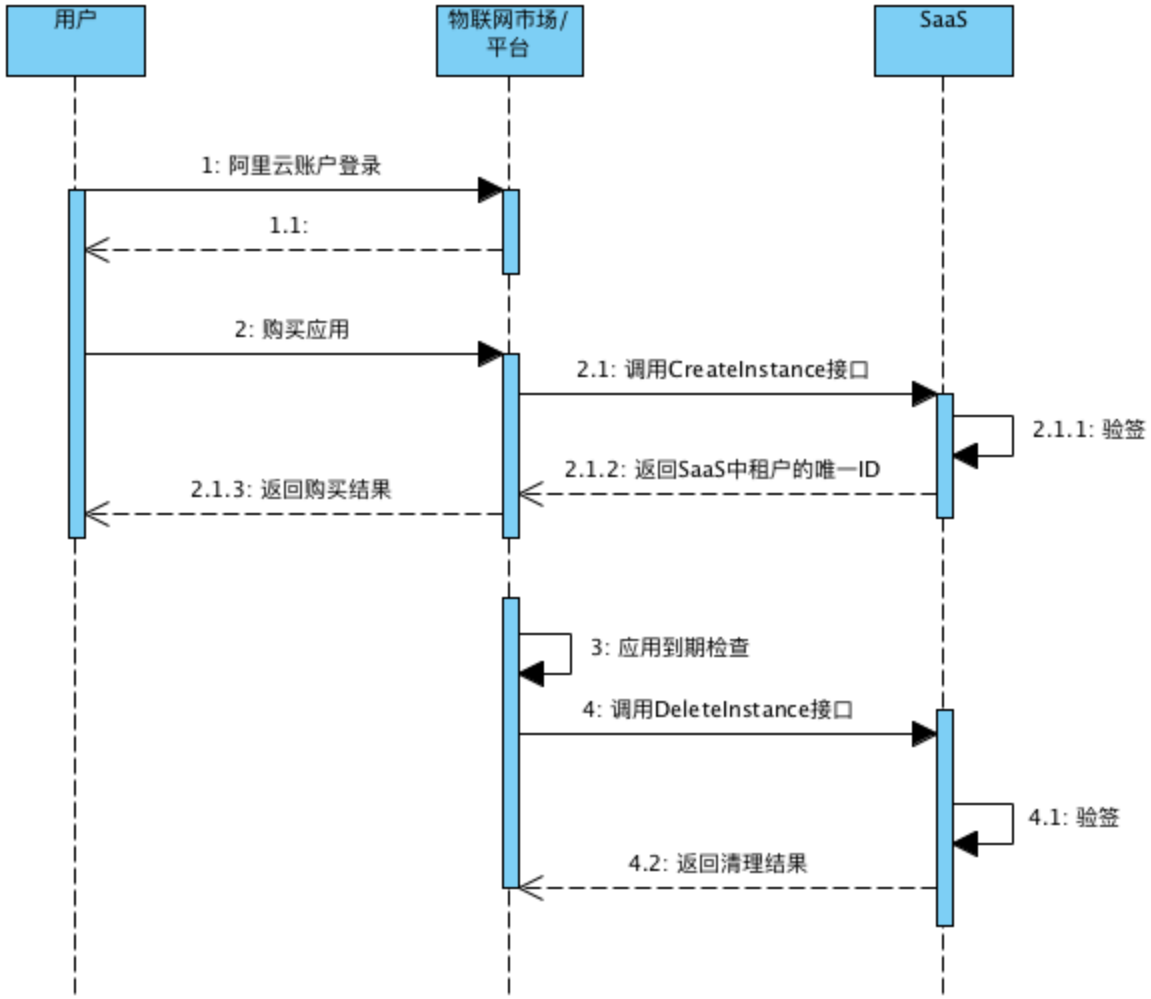
```
{
    "code":200,
    "message":"success",
    "userId":"my saas user id"
}
```

注销租户

场景描述

用户（阿里云账户体系）在物联网市场下单购买应用后需要通知SaaS为当前用户开立一个可访问SaaS服务的租户（账户），用户可以通过点击物联网市场中已购买的应用或者直接访问SaaS提供的公网域名来访问应用，当用户购买的应用到期后系统会自动发起SaaS租户的回收操作。所以您需要实现**DeleteInstance**接口，来实现用户购买应用到期回收场景；

调用时序



接口说明

DeleteInstance注销服务

protocal	httpMethod
HTTPS	POST

请求参数

参数	类型	必填	描述
id	String	是	✓该次访问唯一标示符（幂等验证ID）
tenantId	String	是	IoT平台标识一个租户的唯一ID
userId	String	是	SaaS标识一个租户的唯一ID
appId	String	是	应用唯一ID， 一个租户可以重复购买一款软件， 每次购买appId都不同

返回参数

参数	类型	描述
code	Integer	调用成功返回200； 若调用失败返回203；
message	String	code返回203时填写具体失败原因，code返回200填写success

验签说明

参见【验签说明】

返回示例

```
{
    "code":200,
    "message":"success"
}
```

免密登录

场景描述

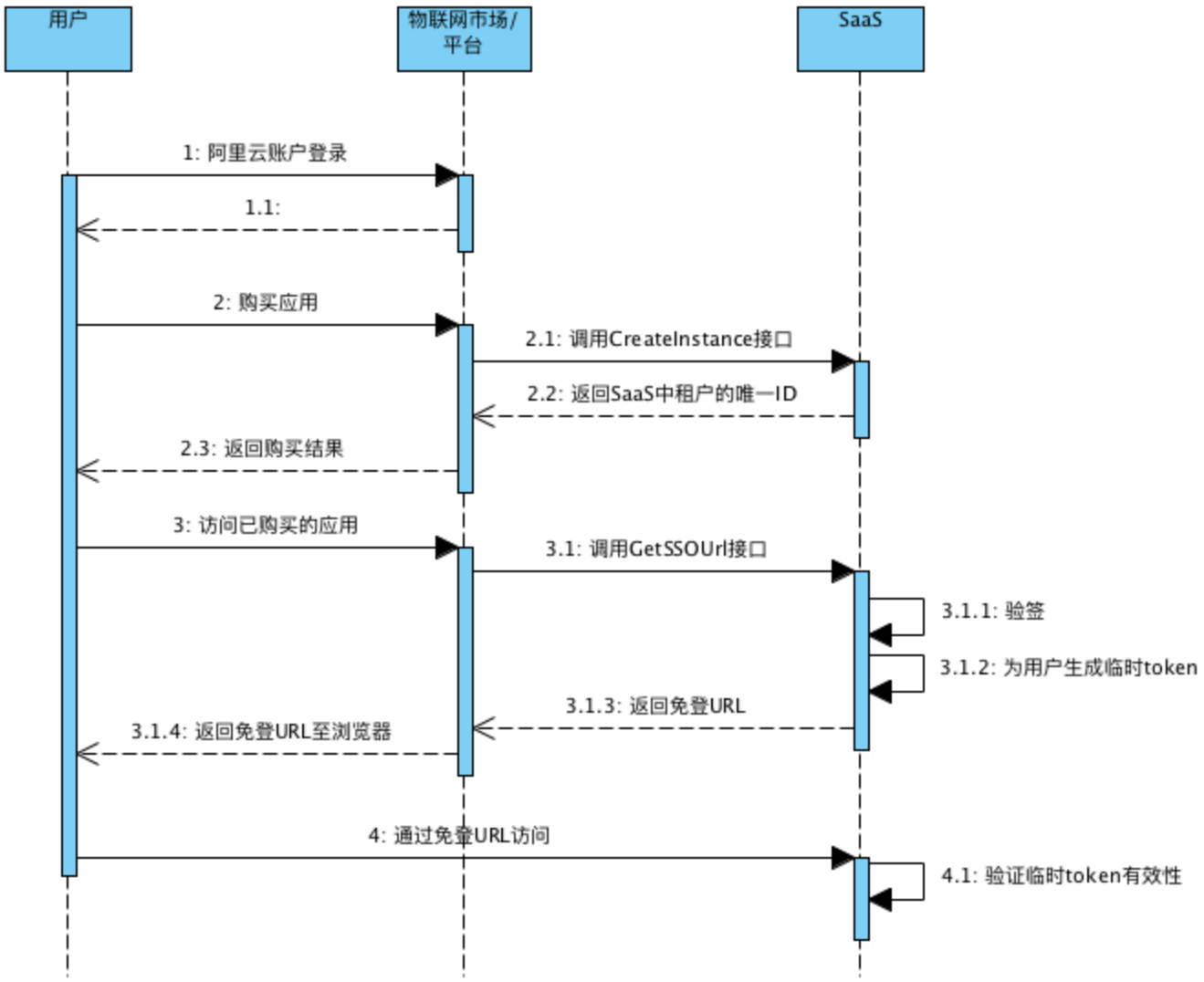
用户（阿里云账户体系）在物联网市场下单购买应用后需要通知SaaS为当前用户开立一个可访问SaaS服务的租户（账户），用户可以通过点击物联网市场中已购买的应用来访问应用，这时用户无需再次输入SaaS的账户/密码即可进入SaaS系统进行业务操作，所以您需要实现**GetSSOUrl**接口，来实现用户访问应用时的免登场景；

安全说明

SaaS返回的SSOUrl必须满足以下两种安全策略中的一种

- **每次生成的Url只允许一次使用**
- **限制临时token的有效时间范围（推荐30秒）**

调用时序



注： SaaS生成的token信息需要带有验证用户身份及时限性，生成和验证的过程全部由SaaS完成，IoT只做免登URL的透传；

接口说明

GetSSOUrl生产服务

protocal	httpMethod
HTTPS	POST

请求参数

参数	类型	必填	描述

id	String	是	✓该次访问唯一标示符（幂等验证ID）
tenantId	String	是	IoT平台标识一个租户的唯一ID
tenantSubUserId	String	否	IoT平台租户组织架构中的员工唯一ID，当员工账号免登时填写
appId	String	是	应用唯一ID， 一个租户可以重复购买一款软件， 每次购买appId都不同
userId	String	是	SaaS标识一个租户的唯一ID

返回参数

参数	类型	描述
code	Integer	调用成功返回200； 若调用失败返回203；
message	String	code返回203时填写具体失败原因，code返回200填写success
ssoUrl	String	免登url， 需要带有验证用户身份及时限性的token信息

验签说明

参见【验签说明】

返回示例

```
{
    "code":200,
    "message":"success",
    "ssoUrl":"https://www.test123.com/login.html?ssoToken=xdasfdasdfasfdasfdaf&checkToken=ddddddd"}
}
```

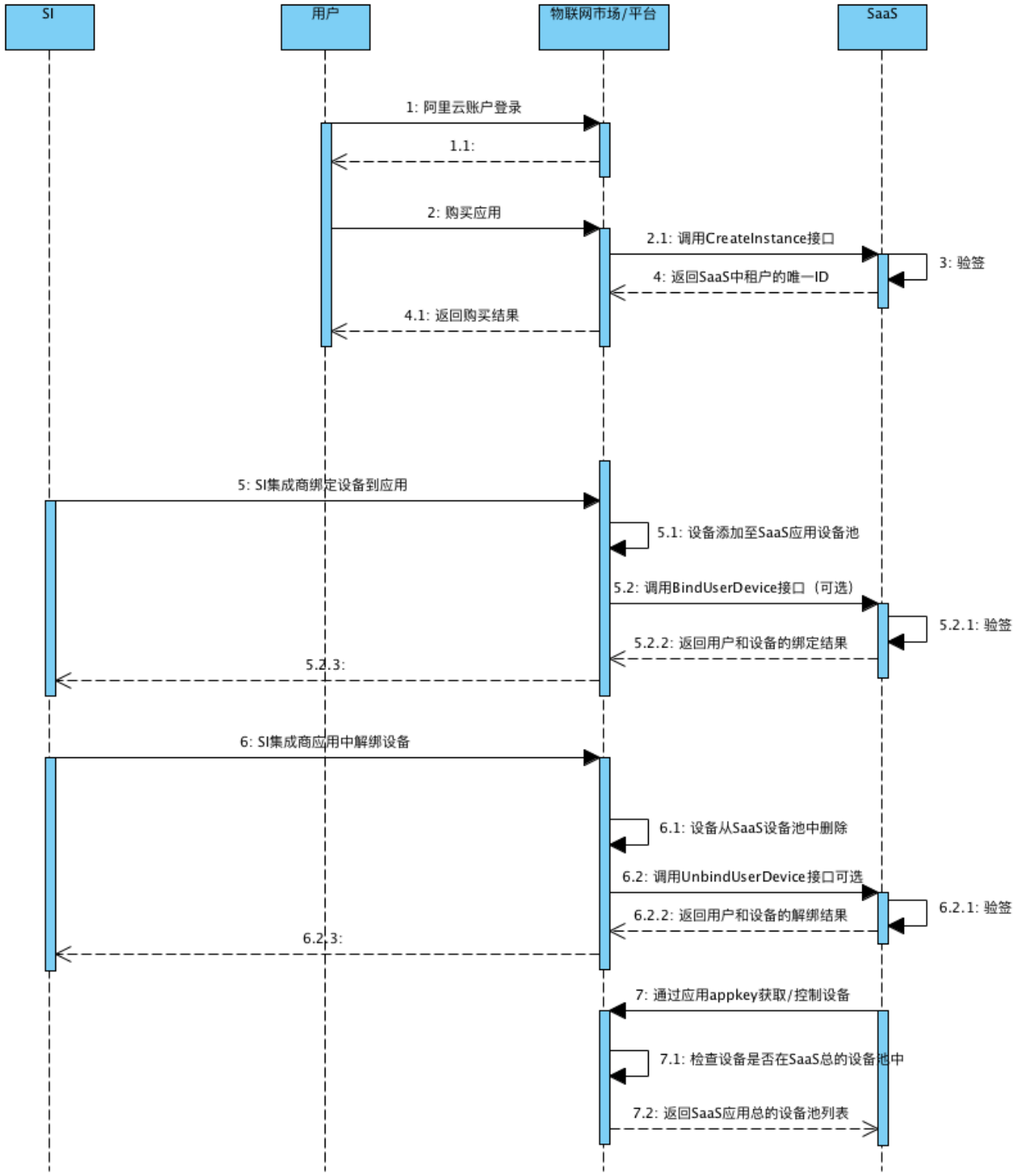
绑定设备(非必须)

场景描述

用户（阿里云账户体系）在物联网市场下单购买应用后需要SI集成商做硬件的施工安装及调试，SI集成商会帮助用户管理应用访问的设备范围，所以每当SI绑定或者解绑设备时，都会通知到SaaS哪个用户绑定或者解绑了哪些设备，这样应用内部无需在开发功能来管理设备和租户的关系，您需要实现**BindUserDevice**接口，来实现SI绑定设备的场景；

注意：如果您的SaaS无设备或者无需建立租户和设备的关系，该接口无需实现

调用时序



接口说明

BindUserDevice解绑设备

protocal	httpMethod
HTTPS	POST

请求参数

参数	类型	必填	描述
id	String	是	✓该次访问唯一标示符（幂等验证ID）
tenantId	String	是	IoT平台标识一个租户的唯一ID
appId	String	是	应用唯一ID， 一个租户可以重复购买一款软件， 每次购买appId都不同
userId	String	是	SaaS标识一个租户的唯一ID
deviceList	Array	是	需要绑定的设备列表， [”pk1:dn1”,”pk2:dn2”,”pk2:dn3”]

返回参数

参数	类型	描述
code	Integer	调用成功返回200； 若调用失败返回203；
message	String	code返回203时填写具体失败原因，code返回200填写success

验签说明

参见【验签说明】

返回示例

```
{
    "code":200,
    "message":"success"
}
```

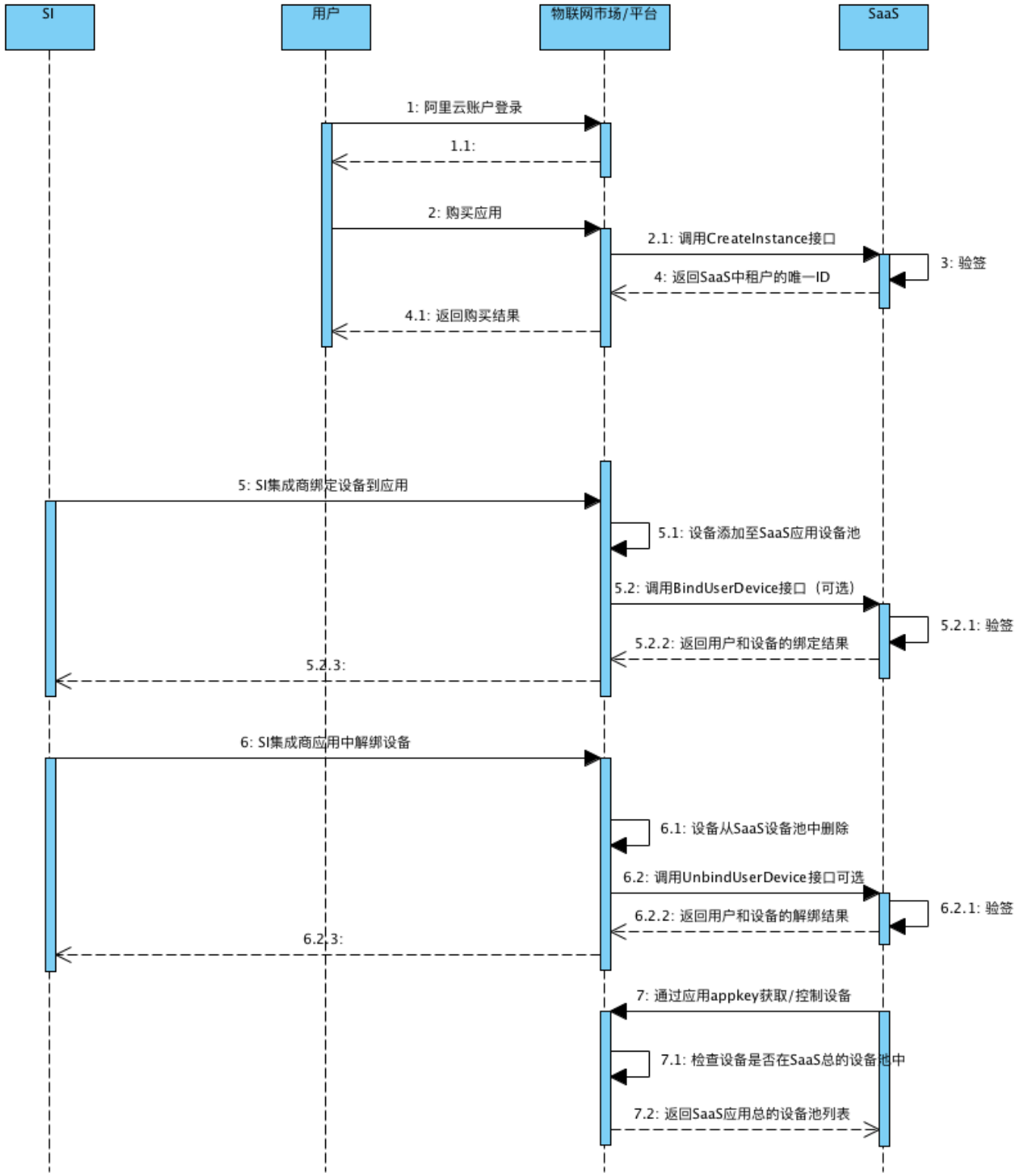
解绑设备(非必须)

场景描述

用户（阿里云账户体系）在物联网市场下单购买应用后需要SI集成商做硬件的施工安装及调试，SI集成商会帮助用户管理应用访问的设备范围，所以每当SI绑定或者解绑设备时，都会通知到SaaS哪个用户绑定或者解绑了哪些设备，这样应用内部无需在开发功能来管理设备和租户的关系，您需要实现**UnbindUserDevice**接口，来实现SI解绑设备的场景；

注意：如果您的SaaS无设备或者无需建立租户和设备的关系，该接口无需实现

调用时序



接口说明

UnbindUserDevice解绑设备

protocal	httpMethod
HTTPS	POST

请求参数

参数	类型	必填	描述
id	String	是	✓该次访问唯一标示符（幂等验证ID）
tenantId	String	是	IoT平台标识一个租户的唯一ID
appId	String	是	应用唯一ID， 一个租户可以重复购买一款软件， 每次购买appId都不同
userId	String	是	SaaS标识一个租户的唯一ID
deviceList	Array	是	需要解绑的设备列表， [”pk1:dn1”,”pk2:dn2”,”pk2:dn3”]

返回参数

参数	类型	描述
code	Integer	调用成功返回200； 若调用失败返回203；
message	String	code返回203时填写具体失败原因，code返回200填写success

验签说明

参见【验签说明】

返回示例

```
{
    "code":200,
    "message":"success"
}
```

调用说明

本章将为您介绍如何调用IoT服务端API；

调用示例

```
public static void main(String[] args) throws Exception {

    SyncApiClient syncClient = SyncApiClient.newBuilder()
                                .appKey("您的AppKey")
                                .appSecret("您的AppSecret")
                                .build();

    IoTApiRequest request = new IoTApiRequest();
    //设置协议版本号
    request.setVersion("1.0");
    String uuid = UUID.randomUUID().toString();
    String id = uuid.replace("-", "");
    //设置请求ID
    request.setId(id);
    //设置API版本号
    request.setApiVer("1.0.0");
    //以下为具体API接口的业务参数设置。API 业务参数请参见各API接口文档。
    request.putParam("tenantId", "xxx");
    request.putParam("appId", "xxx");
    request.putParam("tenantSubUserId", "xxx");
    request.putParam("userId", "xxx");
    //设置请求参数域名、path、request ，如果使用HTTPS，设置为true
    ApiResponse response = syncClient.postBody("api.link.aliyun.com", "/app/user/info/get", request, true);

    System.out.println( "response code = " + response.getStatusCode() + " response content = " + new String(response.getBody(), "utf-8")); "utf-8"));
}
```

```python
# -*- coding: utf-8 -*-
from com.aliyun.api.gateway.sdk
import client from com.aliyun.api.gateway.sdk.http
import request from com.aliyun.api.gateway.sdk.common
import constant
import json

host = "https://api.link.aliyun.com"
url = "/app/user/info/get"

cli = client.DefaultClient(app_key="您的AppKey", app_secret="您的AppSecret")

# post body stream
req_post = request.Request(host=host, protocol=constant.HTTPS, url=url, method
="POST", time_out=30000)

body = {"request": {"apiVer": "1.0.0"}, "params": {"tenantId": "xxx", "appId"
: "xxx","tenantSubUserId": "xxx", "userId": "xxx"}}

req_post.set_body(bytearray(source=json.dumps(body), encoding="utf8"))
req_post.set_content_type(constant.CONTENT_TYPE_STREAM)

print cli.execute(req_post)
```

SDK DEMO

参见【加签demo】

手机获取

场景描述

SaaS服务如果需要租户或者组织架构中员工手机号，可以通过该接口获取；

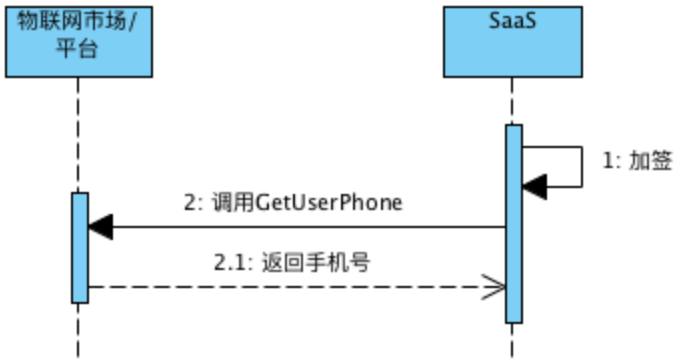
授权过程： IoT平台授权appkey调用该接口

权限展示：该权限属于用户隐私数据，所以用户在市场购买时会提示用户当前应用会读取他的手机，并且SaaS不可泄露用户手机号信息；

安全说明

处于安全考虑，单用户手机号只允许获取一次，如果有业务需求请获取后存储于您的系统中，并在相关性协议条款允许范围内使用该信息；

调用时序



接口说明

GetUserPhone手机号获取服务

protocal	httpMethod	apiVer	url	host
HTTPS	POST	1.0.0	/app/user/info/get	api.link.aliyun.com

请求参数

参数	类型	必填	描述
tenantId	String	是	IoT平台标识一个租户的唯一ID
appId	String	是	应用唯一ID， 一个租户可以重复购买一款软件， 每次购买appId都不同
tenantSubUserId	String	否	IoT平台租户组织架构中的员工唯一ID，当员工账号免登时填写
userId	String	是	SaaS返回的租户唯一ID

返回参数

参数	类型	描述
id	String	IoT平台返回的调用唯一ID
code	Integer	调用成功返回200； 若调用失败返回错误码；
message	String	code返回非200时填写具体失败原因， code返回200填写success

data	”data”:{
	
	”phone”:”13000000000”
	}
		当tenantSubUserId未填写时返回租户tenantId对应手机号，
		当tenantSubUserId填写时返回租户下组织架构中tenantSubUserId的手机号

加签说明

参见【验签说明】

返回示例

```
{
    "id":"dj182318jjkhsljkdhfjahjdhfla"
    "code":200,
    "message":"success",
```

```
    "data":{
        "phone":"13000000000"
    }
}
```

错误码列表

错误码	错误信息	描述
200	success	成功
400	request error	请求错误
401	request auth error	请求认证错
403	request forbidden	请求被禁止
404	service not found	服务未找到
429	too many requests	太多请求
460	request parameter error	请求参数错误
500	service error	服务端错误
503	service not available	服务不可用

接口注册

您接口开发后，需要将您的服务暴露至公网并提交接口注册说明至ISV技术对接群；

需提交的格式说明：

API名称	域名	协议	资源名称（URI）
CreateInstance	例如：www.aliyun.com	http/https	例如：/instance/create
DeleteInstance			
GetSSOUrl			
BindUserDevice			
UnbindUserDevice			

联调测试

接口注册成功后，IoT平台可发起模拟调用，在ISV技术对接群中@支持同学来进行联调测试；

验签说明

以下将为您介绍如何通过AppKey&AppSecret来对HTTP/HTTPS请求做加签，验签的过程需要您按照以下加签过程重新加签，如果您计算的签名和请求的签名一致即可认为验签成功；

1. 签名内容

```
String stringToSign=
HTTPMethod + "\n" +
Accept + "\n" + //建议显示设置 Accept Header。当 Accept 为空时，部分 Http 客户端会给 Accept 设置默认值为 /，导致签名校验失败。
Content-MD5 + "\n"
Content-Type + "\n" +
Date + "\n" +
Headers +
Url
```

HTTPMethod 为全大写，如 POST。
Accept、Content-MD5、Content-Type、Date 如果为空也需要添加换行符”\n”，Headers如果为空不需要添加”\n”。

1.1. Content-MD5

Content-MD5 是指 Body 的 MD5 值，只有当 Body 非 Form 表单时才计算 MD5，计算方式为：

```
// bodyStream 为字节数组。
String content-MD5 = Base64.encodeBase64(MD5(bodyStream.getbytes("UTF-8")));
```

1.2. Headers

Headers 是指参与 Headers 签名计算的 Header 的 Key、Value 拼接的字符串，建议对 X-Ca 开头以及自定义 Header 计算签名，注意如下参数不参与 Headers 签名计算：X-Ca-Signature、X-Ca-Signature-Headers、Accept、Content-MD5、Content-Type、Date。

Headers 组织方法： 先对参与 Headers 签名计算的 Header的Key 按照字典排序后使用如下方式拼接，如果某个 Header 的 Value 为空，则使用 HeaderKey + “:” + “\n”参与签名，需要保留 Key 和英

文冒号。

```
String headers =
HeaderKey1 + ":" + HeaderValue1 + "\n"+
HeaderKey2 + ":" + HeaderValue2 + "\n"+
...
HeaderKeyN + ":" + HeaderValueN + "\n"
```

将 Headers 签名中 Header 的 Key 使用英文逗号分割放到 Request 的 Header 中，Key为：X-Ca-Signature-Headers。

1.3. Url

Url 指 Path + Query + Body 中 Form 参数，组织方法：对 Query+Form 参数按照字典对 Key 进行排序后按照如下方法拼接，如果 Query 或 Form 参数为空，则 Url = Path，不需要添加 ？，如果某个参数的 Value 为空只保留 Key 参与签名，等号不需要再加入签名。

```
String url =
Path +
"?" +
Key1 + "=" + Value1 +
"&" + Key2 + "=" + Value2 +
...
"&" + KeyN + "=" + ValueN
```

注意这里Query或Form参数的Value可能有多个，多个的时候只取第一个 Value 参与签名计算。

2. 计算签名

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");
// secret为AppSecret
byte[] keyBytes = secret.getBytes("UTF-8");
hmacSha256.init(new SecretKeySpec(keyBytes, 0, keyBytes.length, "HmacSHA256"
));
String sign = new String(Base64.encodeBase64(hmacSha256.doFinal(stringToSign.g
etBytes("UTF-8")),"UTF-8"));
```

3. 传递签名

将计算的签名结果放到 Request 的 Header 中，Key为：X-Ca-Signature。

4. 签名排查

当签名校验失败时，API网关会将服务端的 StringToSign 放到 HTTP Response 的 Header 中返回到客户端，Key为：X-Ca-Error-Message，只需要将本地计算的 StringToSign 与服务端返回的 StringToSign 进行对比即可找到问题；
如果服务端与客户端的 StringToSign 一致请检查用于签名计算的密钥是否正确；
因为 HTTP Header 中无法表示换行，因此 StringToSign 中的换行符都被过滤掉了，对比时请忽略换行符。

加签DEMO

您可以直接下载签名的多语言demo，您可以在demo代码中找到加签的逻辑，如果一下列表不包含您的开发语言，您也可以通过阅读【加签说明】来自行开发加签逻辑；

开发语言	sdk demo地址
java	https://github.com/aliyun/iotx-api-gateway-client <dependency> <groupId>com.aliyun.api.gateway</groupId> <artifactId>sdk-core-java</artifactId> <version>1.1.0</version> </dependency>
python	https://github.com/aliyun/api-gateway-demo-sign-python
php	https://github.com/aliyun/api-gateway-demo-sign-php
c#	https://github.com/aliyun/api-gateway-demo-sign-net
android	https://github.com/aliyun/api-gateway-demo-sign-android

BTW，java代码转ApiRequest

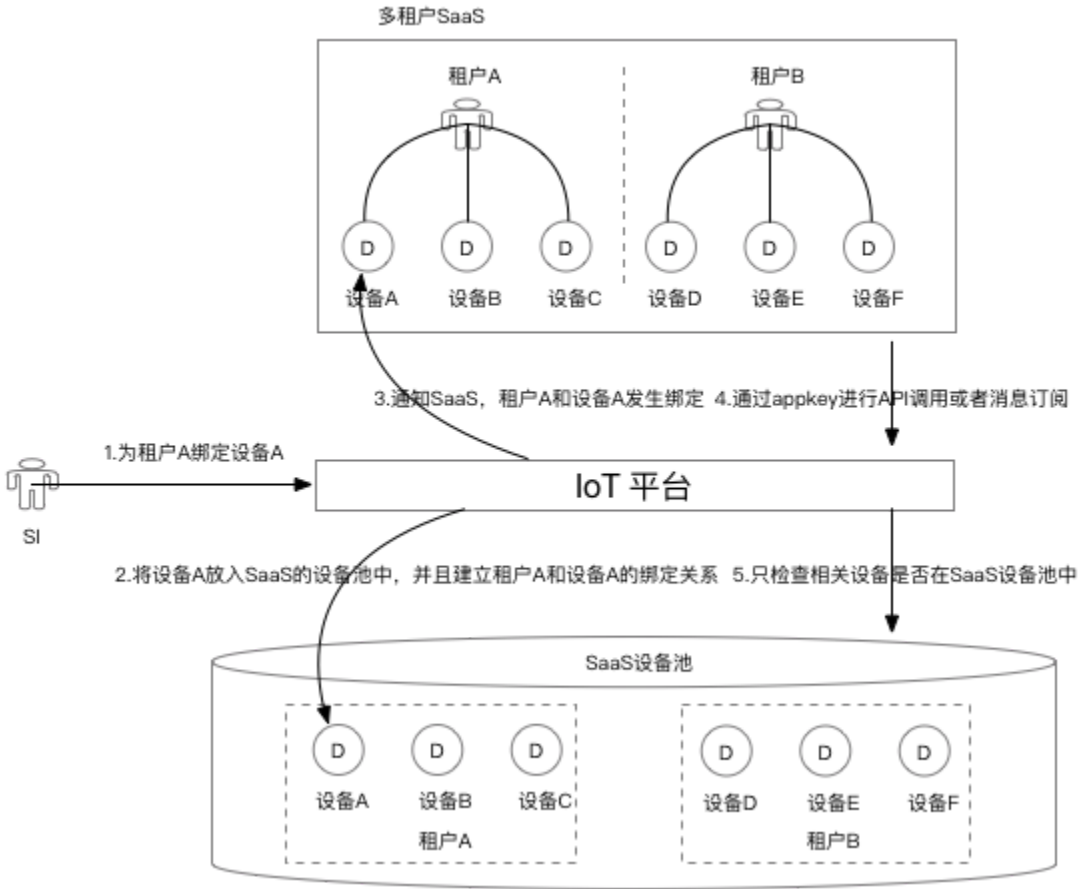
```
private ApiRequest toApiRequest(HttpServletRequest request){
    ApiRequest apiRequest = new ApiRequest(HttpMethod.POST_FORM, request.getRequestURI());
    //验签
    String signHeaders = request.getHeader("X-Ca-Signature-Headers");
    List<String> signHeaderList = new ArrayList<>(16);
    if (StringUtils.isNotBlank(signHeaders)) {
        signHeaderList = Arrays.asList(signHeaders.split(","));
        for (String headerName : signHeaderList) {
```

```
            if (StringUtils.isNotBlank(headerName)) {
                apiRequest.addHeader(headerName, request.getHeader(headerName
));
            }
        }
    }
    apiRequest.addHeader("Accept", request.getHeader("Accept"));
    apiRequest.addHeader("Content-MD5", request.getHeader("Content-MD5"));
    apiRequest.addHeader("Content-Type", request.getHeader("Content-Type"));
    apiRequest.addHeader("Date", request.getHeader("Date"));

    Map<String, String> queryParams = new HashMap<>(8);
    Map<String, String[]> requestParams = request.getParameterMap();
    if (requestParams.size() > 0) {
        for (Map.Entry<String, String[]> param : requestParams.entrySet()) {
            queryParams.put(param.getKey(), param.getValue().length > 0 ? pa
ram.getValue()[0] : "");
            apiRequest.addParam(param.getKey(), param.getValue().length > 0
? param.getValue()[0] : "",
                ParamPosition.QUERY, true);
        }
    }

    return apiRequest;
}
```

附录：未编目文档

设备打通

设备打通



管理边界

- SaaS管理租户和设备之间的权限关系
- IoT平台管理SaaS和设备之间的权限关系

注意：

- 通知SaaS绑定/解绑设备请查看【对接流程.接口开发.绑定设备/解绑设备】
- 应用如何使用设备API或者订阅消息查看https://help.aliyun.com/document_detail/93223.html