统一SDK设计文档

基本特性

1) 提供HTTP/TCP两种接口

2) 服务名称, 字符串, 不使用ID当服务名称

3) 请求参数, json, 这样在HTTP和TCP上面都可以传输, 方便接口的封装, 避免客户端(环境)的复杂性

4) json协议预留校验功能, 可以通过appid对应的私钥计算hash值, 防止伪造参数

5) HTTP协议提供异步接口(不提供同步接口, 支持重传次数), TCP提供发消息功能(异步, 不提供ack)

应用管理平台

建立应用管理平台，任何服务都需要通过管理平台创建基本的应用信息后，服务端从管理平台获取应用的基本信息，并授权客户端接入服务。

App基本信息：

属性	说明
AppName	服务名称, 字符串, 不使用ID当服务名称
AppID	系统统一分配，全局唯一
AppSecret	AppKey，用于数据签名或加密
Desc	应用基本介绍

AppId 及Key

为方便对接入的服务进行统一的统计，管理，便于更好的对所提供的服务进行跟踪、统计，我们给每一个接入方增加一个appId及key。 任何客户端接入服务时需要通过平台注册相应的App信息。包括对应的接口人员，维护跟踪信息等。

传输协议

Http 服务

对于单向服务，建议使用http协议方式提供服务，数据内容统一为json串。 服务请求接口通过url 及参数表示，如：/api/word/filter?work=xxxxx

> {code:0,msg:"",data:{} } // code表示返回值，默认0表示成功，msg表示对应错误码等文本信息，data表示返回的数据，可以为json对象或数组，依据具体业务而定。

Https支持

服务器和服务器之间无需协议的加解密操作，对于公网服务，可以通过后台添加白名单的机制进行安全保障，对于给客户端提供的服务，可以增加https服务保障服务安全。

TCP流服务

特殊情况下建议使用TCP流方式接入，统一消息头部。

> |———len———|———type———| //len为4字节长度，type表示消息类型4字节长度

协议代理网关

开发通用的协议网关，简化客户端协议的处理，网关完成后端服务协议的转化，也保障公网服务的安全。代理网关完成协议的转化，建议对客户端的SDK采用此方式。

> client =======> gateway （协议转化）========> service

编程语言

主要提供c++ sdk包，包括tcp 流客户端，http客户端支持。 尽可能少使用第三方库或尽量考虑源码依赖库，支持跨平台（linux/windows)平台。 可以提供源码包方式。

DNS解析

在实现跨服务sdk中，比较难处理的问题在于DNS解析过程，DNS解析受制于用户的网络情况，异常情况下可能出现分钟级别的等待时间，服务器之间尽量不使用DNS解析，SDK库应该支持使用宏关闭DNS解析功能。

同步vs异步

同步接口使用简单，服务器要实现类rpc的调用方式，无需关心底层协议细节，sdk实现请求和相应的一一对应。但同步接口在等待响应时可能会比较消耗时间，耗时比较长，影响业务层对性能。

异步方式通过单独的线程发送服务请求，通过队列的方式接收响应。 需要应用层记录请求和响应的对应关系。 建议采用异步的方式进行业务消息的回调处理，减少对业务层的影响。

加密&签名

常用加密算法

名称	数据大小(MB)	时间(s)	平均速度MB/S	评价
DES	256	10.5	22.5	低
3DES	256	12	12	低
AES(256-bit)	256	5	51.2	中
Blowfish	256	3.7	64	高

散列算法

名称	安全性	速度
SHA-1	高	慢
MD5	中	快

> BLOWFISH，它使用变长的密钥，长度可达448位，运行速度很快；
>
> TEA*(Tiny Encryption Algorithm)*简单高效的加密算法，加密解密速度快，实现简单。但安全性不如DES，TX一直用tea加密
>
> 参考：各种加密算法比较

对于加密的内容，我们采用blowfish进行加密，速度较快，安全性较高，对于签名算法我们使用MD5方式进行签名。

签名步骤：

①接口提供方给出appid和appsecret

②调用方根据appid和appsecret以及请求参数，按照一定算法生成签名sign

③接口提供方验证签名

生成签名步骤：

①将所有业务请求参数按字母先后顺序排序

②参数名称和参数值链接成一个字符串A

③在字符串A的首尾加上appsecret组成一个新字符串B

④对字符串进行md5得到签名sign

配置

SDK的配置信息都需要通过初始化接口获取配置，不依赖配置文件，用户可以将配置信息写入自己的配置文件中，SDK包不负责维护配置文件存储位置，读写操作等。 SDK包的配置都是动态的，程序重启后需要重新初始化。 SDK可以提供reload功能重新清空内部运行状态以获得最佳性能表现或解决可能存在隐藏问题。

日志

SDK内不提供写日志功能，需要应用层注册日志写入函数，SDK内部进行业务处理时会调用应用层注入的回调函数写日志。 SDK定义通用的日志级别：debug/info/warning/fatal等，应用层可以动态打开或关闭日志或设置日志级别。

SDK编码标准

SDK编码尽可能保证接口简单易用，接口调用顺序可以要求强一致性，如果不一致要有明显错误提示或告警，内部不建议使用断言。

对用户传入的任何参数都要做合法性检查，包括空指针检查，数值范围检查等。

接口返回值清新明了，统一0表示成功，负值表示错误，具体错误码在头文件中定义说明清楚。

版本管理

作为服务SDK包，版本的稳定至关重要，SDK以源码方式提供，由技术中心统一维护管理，内部开源方式公开，提供不同平台的编译脚本。

任何一次提交项目组使用，SDK都需要做好对应使用版本记录。