|  |  |
| --- | --- |
| ICS |  |
| CCS | 点击此处添加CCS号 |

|  |
| --- |
|  |

     地方标准

DB XX/T XXXX—XXXX

智慧文旅信息系统数据接口规范

点击此处添加标准名称的英文译名

（本草案完成时间：11.26）

XXXX - XX - XX发布

XXXX - XX - XX实施

       发布

目次

[1 范围 1](#_Toc151968346)

[2 规范性引用文件 1](#_Toc151968347)

[3 术语和定义 1](#_Toc151968348)

[4 接口技术要求 1](#_Toc151968349)

[5 政务外网数据接口 2](#_Toc151968350)

[6 公网数据接口 9](#_Toc151968351)

[附录A（规范性） 政务外网Digest认证示例 12](#_Toc151968352)

[附录B（规范性） 公网认证示例 18](#_Toc151968353)

智慧文旅信息系统数据接口规范

* 1. 范围

本文件规定了智慧文旅信息系统的数据接口技术要求、政务外网数据接口、公网数据接口等要求。

本文件适用于智慧文旅信息系统与政务机构相关部门、外部第三方组织机构不同权限接口的应用和管理。

* 1. 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

DB43/T 1754-2020 政务大数据中心数据交换规范

RFC 2616 超文本传输协议1.1 （Hypertext transfer protocol – HTTP/1.1）

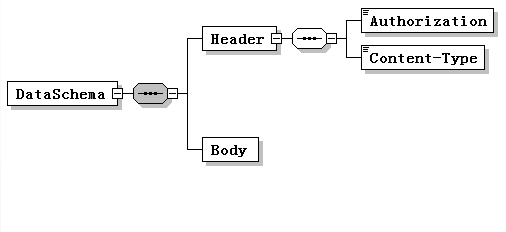
RFC 4627 JSON格式定义（The application/JSON media type for javascript object notation）

* 1. 术语和定义

本文件没有需要界定的术语和定义。

* 1. 接口技术要求
     1. 接口模型

数据接口模型用于信息交换传输时封装交换信息。数据接口模型的数据文档结构名称是DataSchema，由接口数据头（Header）信息、数据体（Body）组成。接口Header是必选元素，用来描述数据申请方的身份信息（Authiorization）和内容类型信息（Content-Type）。Body是必选元素，用来封装交换的数据信息，Body的封装要求应符合DB43/T 1754-2020第6.2条的要求。接口模型如图1所示。



* + 1. 接口流程

数据接口应使用基于HTTP/HTTPS协议的JSON格式的消息体传输数据。流程使用了“请求-Request”、“指示-Indication”、“响应-Response”、“确认-Confirm”四条服务原语，流程见图2。



1. 接口服务流程
   * 1. 接口调用

接口调用方式包括：

1. 请求：请求接口应按照RFC 2616中规定的要求，使用Get、Post、Put、Delete等方法调用平台数据服务接口，请求接口将请求参数按照JSON格式封装后，在HTTP的Request消息中发送给平台方；
2. 响应：请求接口调用成功或失败，平台应将处理结果按照JSON格式封装后，放置在HTTP的Response消息中，返回给调用者。HTTP的Response消息应符合RFC 2616中的要求。
   * 1. 接口基本要求

接口应满足以下条件：

1. 采用UTF-8字符编码；
2. 支持数据传输过程中的机密性与完整性保护；
3. 接口数据应符合RFC 4627中JSON格式要求。
   * 1. 接口基本分类

按照数据共享交换的网络不同进行分类区分，可以将接口分为政务外网数据接口、公网数据接口2类：

1. 政务外网数据接口：部署在政务外网上，用于政府机构和相关部门进行数据资源共享，以及按权限对社会进行开放的数据接口。
2. 公网数据接口：部署在公网上，用于与智慧文旅系统进行数据交互的企业、第三方组织和公众的数据资源接口。
   1. 政务外网数据接口
      1. 接口描述

通过电子政务外网提供给政府机构和相关部门的数据接口，通过数据的一致性、可访问性和互操作性，以实现政府部门之间数据交换和共享，支持公共服务决策分析、信息查询以及协同工作等应用场景。

* + 1. 接口方法

接口应遵循HTTP REST风格：

1. GET: 用于从服务器获取数据；
2. POST: 用于向服务器提交数据；
3. PUT: 用于向服务器更新已经存在的资源；
4. DELETE: 用于删除服务器上的资源。
   * 1. 认证和授权
        1. 验证规则如下：
5. 由省交换平台提供唯一的应用码；
6. 由省交换平台提供唯一的授权码用于签名；
7. 由省交换平台归功唯一的SM4密钥，用于数据加解密；
8. 加密算法采用SM4；
9. 摘要算法采用MD5。
   * + 1. 政务外网Digest认证

访问接口时，需要在HTTP请求头中携带"Authorization", "Authorization"的值对应格式（见表1）为：

Algorithm=Algorithm,TimeStamp=TimeStamp,AccessKeyId=AccessKeyId,Signature=Signature

| 名称 | 描述 |
| --- | --- |
| Algorithm | 用于计算签名的算法 |
| TimeStamp | 时间戳，为API调用时刻对应的UTC时间(北京时间-8小时)，时间格式如下"2015-01-01 01:01:01"。   1. TimeStamp不能与服务端时间相差超过5分钟。 |
| Access Key Id | 授权后为用户分配的Access Key ID，服务端使用该字段来校验签名合法性。 |
| Signature | 通过计算得到的签名，参加签名的字段有：URL参数，TimeStamp, Access Key Secret。签名过程如下：  1.URL参数规范化  1).以&符号为分隔符，得到参数列表（不包含#号后面部分）。  2).把分隔后得到的字符串按参数名称的字典顺序依次使用&符号连接。  3).使用UTF-8字符集进行URL编码。  2.构造用于计算签名的字符串  StringToSign=  HTTPMethod + "&" +  percentEncode("/") + "&" +  percentEncode("TimeStamp") + "&" +  URL参数规范化的值  其中percentEncode指使用UTF-8字符集进行编码。  3.计算签名  使用Algorithm对StringToSign进行签名，签名密钥为AccessKeyId对应的AccessKeySecret。 |

1. 认证算法详见附录A。
   * 1. 接口内容

接口内容包括但不限于：

1. 景区基础数据
2. 景区客流数据
3. 景区视频数据
4. 图书馆基础数据
5. 图书馆客流数据
6. 图书馆视频数据
7. 文化馆基础数据
8. 文化馆客流数据
9. 文化馆视频数据
10. 博物馆基础数据
11. 博物馆客流数据
12. 博物馆视频数据
13. 旅行社数据
14. 导游数据
15. 旅游团组数据
    * 1. 接口约束

接口约束主要如下：

1. 响应时间 <= 30秒；
2. 单个文件传输大小 <= 10M。
   * 1. 消息状态码

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| http状态码 | 描述 | code | 说明 | 备注 |
| 200 | 请求成功 | 0 | 请求成功 | 处理返回数据 |
| 1 | 系统内部错误 |  |
| 2 | HTTP解析错误 |  |
| 3 | 结果解析错误 |  |
| 4 | 用户认证错误 | 根据提示检查用户信息 |
| 12 | 参数不正确 | 根据提示检查参数 |
| 13 | 参数不能为空 | 根据提示检查参数 |
| 14 | 参数类型不正确 | 根据提示检查参数 |
| 15 | 无权限访问 | 根据提示检查参数 |
| 16 | 参数不存在 | 根据提示检查参数 |
| 17 | URL格式错误 | 根据提示检查参数 |
| 18 | 参数不能为空 | 根据提示检查参数 |
| 19 | 参数长度不能超过参数 | 根据提示检查参数 |
| 20 | 参数长度不能小于参数 | 根据提示检查参数 |
| 21 | 参数不能含有特殊字符 | 根据提示检查参数 |
| 22 | 参数不能含有中文字符 | 根据提示检查参数 |
| 23 | 参数地址格式不正确 | 根据提示检查参数 |
| 24 | 参数不是数字 | 根据提示检查参数 |
| 25 | 参数校验配置错误 | 根据提示检查参数 |
| 26 | 参数参数格式不正确 | 根据提示检查参数 |
| 27 | 代理服务授权类型不正确 | 根据提示检查 |
| 28 | 时间格式不正确 | 根据提示检查 |
| 30 | 当前路由规则不存在 | 根据提示检查 |
| 31 | 未知的路由类型 | 根据提示检查 |
| 32 | 当前用户尚未进行应用申请 | 根据提示检查 |
| 40 | 当前应用不存在 | 根据提示检查 |
| 41 | AccessKeyId已存在 | 根据提示检查 |
| 42 | Username已存在 | 根据提示检查 |
| 43 | AccessKeyId与Username不能相同 | 根据提示检查 |
| 44 | AccessKeyId不能被修改 | 根据提示检查 |
| 45 | Username不能被修改 | 根据提示检查 |
| 46 | 远程路由%s创建应用[%s]失败 | 根据提示检查 |
| 47 | 远程路由%s删除应用[%s]失败 | 根据提示检查 |
| 48 | 远程路由%s更新应用[%s]失败 | 根据提示检查 |
| 49 | 当前应用已经存在 | 根据提示检查 |
| 50 | 当前应用名称已经存在 | 根据提示检查 |
| 51 | 当前应用与其它用户冲突 | 根据提示检查 |
| 52 | 获取应用失败 | 根据提示检查 |
| 53 | 创建应用失败 | 根据提示检查 |
| 54 | 删除应用失败 | 根据提示检查 |
| 60 | 当前路由授权信息不存在 | 根据提示检查 |
| 61 | 远程路由%s执行路由授权[%s]失败 | 根据提示检查 |
| 62 | 远程路由%s取消路由授权[%s]失败 | 根据提示检查 |
| 63 | 远程路由%s获取路由授权[%s]失败 | 根据提示检查 |
| 84 | 已经对当前路由进行过授权 | 根据提示检查 |
| 65 | 获取路由授权失败 | 根据提示检查 |
| 70 | 当前路由与实例不匹配 | 根据提示检查 |
| 80 | 计算签名时发生错误 | 根据提示检查 |
| 81 | 实例名称已经存在 | 根据提示检查 |
| 90 | 数据转换错误 | 根据提示检查 |
| 91 | 参数验证错误 | 根据提示检查 |
| 92 | 路由注册到ZooKeeper错误 | 根据提示检查 |
| 93 | 路由从ZooKeeper删除错误 | 根据提示检查 |
| 100 | 数据转换错误 | 根据提示检查 |
| 101 | ZooKeeper连接错误 | 根据提示检查 |
| 102 | ZooKeeper内部错误 | 根据提示检查 |
| 110 | 用户获取失败 | 根据提示检查 |
| 111 | 登录用户不存在 | 根据提示检查 |
| 112 | 没有权限，拒绝操作 | 根据提示检查 |
| 113 | 数据源不存在 | 根据提示检查 |
| 114 | 数据源不存在 | 根据提示检查 |
| 115 | 数据源被使用，无法修改或删除。 | 根据提示检查 |
| 116 | 生成SQL失败 | 根据提示检查 |
| 121 | 创建失败:路由类型错误。 | 根据提示检查 |
| 122 | 创建失败： | 根据提示检查 |
| 123 | 操作失败：[%s] | 根据提示检查 |
| 124 | 删除路由失败。 | 根据提示检查 |
| 124 | 路由信息为空 | 根据提示检查 |
| 124 | 删除路由失败,该服务已被订阅。 | 根据提示检查 |
| 130 | Swagger文件类型错误 | 根据提示检查 |
| 131 | Swagger文件内容解析错误 | 根据提示检查 |
| 132 | 读取文件内容失败 | 根据提示检查 |
| 133 | 文件类型错误 | 根据提示检查 |
| 140 | 订购失败 | 根据提示检查 |
| 141 | 订购授权失败 | 根据提示检查 |
| 150 | 启用路由失败 | 根据提示检查 |
| 151 | 停止路由失败 | 根据提示检查 |
| 125 | 尚未配置邮件服务器。 | 根据提示检查 |
| 126 | 创建邮件服务器失败 | 根据提示检查 |
| 127 | 更新邮件服务器失败 | 根据提示检查 |
| 124 | 无法连接的邮件服务器 | 根据提示检查 |
| 128 | 创建邮件发送规则失败 | 根据提示检查 |
| 128 | 已在该网关下配置过发送规则 | 根据提示检查 |
| 129 | 更新邮件发送规则失败 | 根据提示检查 |
| 130 | 处理告警事件失败 | 根据提示检查 |
| 131 | 清空告警事件失败 | 根据提示检查 |
| 132 | 获取告警事件列表失败 | 根据提示检查 |
| 133 | 创建告警规则失败 | 根据提示检查 |
| 134 | 创建告警规则失败 | 根据提示检查 |
| 135 | 获取告警规则失败 | 根据提示检查 |
| 136 | 删除告警规则失败 | 根据提示检查 |
| 46 | 远程路由%s创建应用[%s]失败 | 根据提示检查 |
| 47 | 远程路由%s删除应用[%s]失败 | 根据提示检查 |
| 133 | 告警规则已存在 | 根据提示检查 |
| 140 | SQL输出为空！ | 根据提示检查 |
| 150 | 获取调用记录失败！ | 根据提示检查 |
| 151 | 获取统计信息失败！ | 根据提示检查 |
| 152 | 统计信息为空 | 根据提示检查 |
| 160 | 审核失败。 | 根据提示检查 |
| 170 | 更新失败，当前路由存在运行中的工作流。 | 根据提示检查 |
| 171 | 更新失败，当前路由版本信息错误。 | 根据提示检查 |
| 180 | 请勿重复订阅 | 根据提示检查 |
| 190 | 请勿重复删除 | 根据提示检查 |
| 200 | 服务正在删除，请勿订阅 | 根据提示检查 |
| 201 | 服务删除失败 | 根据提示检查 |
| 202 | 服务更新失败 | 根据提示检查 |
| 203 | 服务保存失败 | 根据提示检查 |
| 210 | 插件名称已存在 | 根据提示检查 |
| 215 | 插件名称不一致 | 根据提示检查 |
| 216 | 插件已经关联服务 | 根据提示检查 |
| 217 | 插件内容为空 | 根据提示检查 |
| 3005 | 文件删除失败 | 根据提示检查 |
| 3006 | 文件创建失败 | 根据提示检查 |
| 3007 | 下载失败，文件不存在 | 根据提示检查 |
| 6000 | HTTP请求出错 | 根据提示检查 |
| 6001 | 服务器内部错误 | 根据提示检查 |
| 6002 | 获取网关信息失败 | 根据提示检查 |
| 6003 | 用户信息获取失败 | 根据提示检查 |
| 118 | 获取当前部门信息失败 | 根据提示检查 |
| 20501 | 类目创建失败 | 根据提示检查 |
| 20502 | 同级中类目名称已经存在 | 根据提示检查 |
| 20503 | 父级类目不存在 | 根据提示检查 |
| 20504 | 父级类目不存在 | 根据提示检查 |
| 20506 | 该类目已经被使用 | 根据提示检查 |
| 20507 | 该类目下存在子节点 | 根据提示检查 |
| 20508 | 类目删除失败 | 根据提示检查 |
| 20509 | 有子节点,不能删除 | 根据提示检查 |
| 20510 | 获取类目详情信息失败 | 根据提示检查 |
| 20511 | 获取类目树信息失败 | 根据提示检查 |
| 20512 | 无法将不同level排序 | 根据提示检查 |
| 20513 | [%s]条类目创建失败 | 根据提示检查 |
| 20514 | [%s]条类目更新失败 | 根据提示检查 |
| 20515 | [%s]条类目删除失败 | 根据提示检查 |
| 20516 | 类目编码不正确 | 根据提示检查 |
| 20517 | 类目编码%s已存在 | 根据提示检查 |
| 22501 | 类目编码规则创建失败 | 根据提示检查 |
| 22502 | 存在类目，%s级前的类目编码规则不允许修改 | 根据提示检查 |
| 22503 | 类目编码规则修改失败 | 根据提示检查 |
| 22504 | 类目编码规则不存在 | 根据提示检查 |
| 22505 | 类目编码规则删除失败 | 根据提示检查 |
| 22506 | 事权单位不能重复设置 | 根据提示检查 |
| 22507 | 16位类目编目长度之和不能超过128位 | 根据提示检查 |
| 22508 | 该级编目规则下已创建类目，不允许修改 | 根据提示检查 |
| 22509 | 权限不足，创建类目失败 | 根据提示检查 |
| 22509 | 权限不足，修改类目失败 | 根据提示检查 |
| 21201 | 设置背景失败 | 根据提示检查 |
| 20007 | 创建License提示信息失败 | 根据提示检查 |
| 20008 | 部门绑定网关总数目超出License限制 | 根据提示检查 |
| 20009 | 指定的网关不存在 | 根据提示检查 |
| 133 | 告警规则已存在 | 根据提示检查 |
| 140 | SQL输出为空！ | 根据提示检查 |
| 302 | 重定向，为了完成请求，必须进一步执行的动作 | 30201 | 重定向 | 访问返回数据中的链接 |
| 400 | Bad Request 缺失参数，参数格式不正确 | 40001 | 缺少必选参数 | 根据提示检查调用参数 |
| 40002 | 非法的参数 | 根据提示检查调用参数 |
| 401 | Unauthorized 没有授权 | 40101 | 没有授权 | 需要授权，XXXX |
| 404 | Not Found 请求的资源不存在 | NA | NA | NA |

* 1. 公网数据接口
     1. 接口描述

通过公网（internet）提供给企业及第三方组织和公众使用的数据接口，支持数据的实时查询、获取、分享和分析，以满足商业应用、研究分析和社会创新的需求。

* + 1. 接口方法

接口方法为POST。

* + 1. 接口请求参数

请求参数如表1所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 报文名称 | 英文名称 | 可空 | 数据类型 |
| 1 | 账号ID | appId | 否 | String(32) |
| 2 | 报文实体 | bizContent | 否 | String |
| 3 | 接口名称 | name | 否 | String |
| 4 | 请求Id | requestId | 否 | String |
| 5 | 时间戳 | timestamp | 否 | String |
| 6 | 版本号 | version | 否 | String |
| 7 | 签名 | sign | 否 | String |

1. 非空值待签名参数按照首字母以ASCII方式升序排列（ASCII ASC），对于相同字母则使用下个字母做二次排序，字母序为从左到右，以此类推。参数顺序为：appId，bizContent，name, requestId，timestamp，version。
   * 1. 认证和授权
        1. 采用Token认证方式，访问接口时需通过3步完成：
2. 客户端用户注册，获得智慧文旅平台提供的用户名及密码；
3. 客户端通过用户ID（client\_id）和密钥（client\_secret）登录平台，获得Token；
4. 客户端使用Token依权限访问服务器资源。
   * + 1. 验证规则如下：
5. 由智慧文旅平台提供唯一账号和私钥；
6. 签名算法为MD5；
7. Token有效期为2个小时，Token丢失后应重新获取Token。
   * + 1. 公网签名认证

访问接口时，将排序后的接口请求参数按照参数名（key）参数值（value）的次序进行字符串拼接，拼接处不包含任何字符，并在该长串的头部及尾部分别拼接密钥（secret），完成签名字符串的组装，并在MD5加密后做大写转换。如：sign=MD5(appKey+key+value+appKey).toUpperCase()。示例详见附录B。

* + 1. 接口内容

接口内容包括但不限于：

1. 景区门票信息；
2. 酒店信息。
   * 1. 接口约束

接口约束主要如下：

1. 响应时间 <= 20秒；
2. 单个文件传输大小 <= 10M。
   * 1. 消息状态码

公网数据接口的服务状态码如表2：

|  |  |
| --- | --- |
| 规范编码 | 错误信息 |
| 10000 | success |
| 20000 | 系统错误 |
| 20001 | 服务繁忙 |
| 20002 | 超时异常 |
| 21000 | 缺少requestId参数 |
| 21001 | 缺少签名参数 |
| 21002 | 缺少appId参数 |
| 21003 | 缺少时间戳参数 |
| 21004 | 缺少接口名参数 |
| 21005 | 缺少业务参数 |
| 21006 | Content-Type请求头参数必须为application/x-www-form-urlencoded |
| 21007 | 不支持Header的Expect请求头参数 |
| 22000 | 接口路由不存在 |
| 22001 | 接口服务不可用 |
| 23000 | 签名无效 |
| 23001 | 无效的appId参数或未绑定接口权限 |
| 40000 | 业务处理失败 |
| 24000 | 没有当前接口权限 |
| 24001 | 缺少parkCode景区编码参数 |
| 24002 | 请求业务404，请检查接口路径是否正常 |
| 24009 | 限流异常 |
| 26000 | 无效参数 |
| 26001 | 业务接口异常 |
| 26002 | 未知错误 |
| 26003 | bizContent请求报文必须是json格式且不能为空 |
| 26006 | 参数检查不符合预期 |
| 26007 | 业务校验失败 |
| 26008 | HTTP请求错误 |
| 26009 | 接口限流了，请稍后重试 |
| 26010 | 服务降级了，请稍后重试 |
| 26011 | 接口限流了，请稍后重试 |
| 26012 | 触发系统保护规则 |
| 26013 | 授权规则不通过 |

2. （规范性）  
   政务外网Digest认证示例
   1. 认证代理信息

接口服务中的代理信息如表A.1所示：

* 1. 接口服务代理信息

|  |  |
| --- | --- |
| 名称 | 值 |
| URL | http://www.datamall.com/api/v1.0/catlog?id=1&flag=true&type=json |
| HTTPMethod | GET |
| TimeStamp | 2016-01-01 01:01:01 |
| Algorithm | HMAC-SHA256 |
| Access Key Id | bf796c1d7081462a49042c0a71ed9b143 |
| Access Key Secret | 8bf76c1d7081462a9042c0a71ed9b142 |

* 1. Authorition请求头算法

请求算法如下：

import org.apache.commons.codec.binary.Base64;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import java.io.UnsupportedEncodingException;

import java.net.URLDecoder;

import java.net.URLEncoder;

import java.text.SimpleDateFormat;

import java.util.Arrays;

import java.util.Calendar;

import java.util.Map;

import java.util.TreeMap;

/\*\*

\* DataMall签名计算工具类

\* Created by w0592 on 2017/11/14.

\*/

public class SignatureUtils {

public static final String DMALL\_HEAD\_ALGORITHM = "Algorithm";

public static final String DMALL\_ACCESSKEY\_ID = "AccessKeyId";

public static final String DMALL\_HEAD\_SIGNATURE = "Signature";

public static final String DMALL\_TIMESTAMP = "TimeStamp";

public static final String UTF\_8 = "UTF-8";

/\*\*

\* 获取时间戳

\*/

public static String getUTCTime() {

// 1、取得本地时间：

Calendar cal = Calendar.getInstance();

// 2、取得时间偏移量：

int zoneOffset = cal.get(Calendar.ZONE\_OFFSET);

// 3、取得夏令时差：

int dstOffset = cal.get(Calendar.DST\_OFFSET);

// 4、从本地时间里扣除这些差量，即可以取得UTC时间：

cal.add(Calendar.MILLISECOND, -(zoneOffset + dstOffset));

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

return sdf.format(cal.getTime());

}

/\*\*

\* 签名计算方法

\*

\* @param accessKeyId 密钥id

\* @param accessKeySecret 密钥secret

\* @param httpMethod http请求方法类型，目前支持 POST, PUT, DELETE, GET四种

\* @param queryString http请求参数，通常为?号之后的内容

\* @param timeStamp 时间戳, 格式 yyyy-MM-dd HH:mm:ss，注意时区为UTC时区，如何获取utc时间，参看getUTCTime方法

\* 注意：为防止重放，调用方传入的时间与服务器时间不能偏差超过5分钟

\*

\* @return 计算结果，该字段为String类型，上层需要把该字段放到http header中，其中key为Authorization，value为返回结果

\*/

public static String getAuthorization(String accessKeyId, String accessKeySecret, String queryString, String httpMethod, String timeStamp) {

/\* 构造计算签名的数据 \*/

String signature = getSignString(queryString, httpMethod, timeStamp, accessKeySecret);

if (null == signature) {

return null;

}

StringBuilder auth = new StringBuilder();

auth.append(DMALL\_HEAD\_ALGORITHM).append("=").append("HMAC-SHA256").append(",")

.append(DMALL\_ACCESSKEY\_ID).append("=").append(accessKeyId).append(",")

.append(DMALL\_TIMESTAMP).append("=").append(timeStamp).append(",")

.append(DMALL\_HEAD\_SIGNATURE).append("=").append(signature);

return auth.toString();

}

private static String getSignString(String queryString, String httpMethod, String timeStamp, String accessKeySecret) {

String strToSign = null;

try {

if (null != queryString) {

queryString = URLDecoder.decode(queryString, UTF\_8);

}

strToSign = buildCanonicalString(queryString, httpMethod, timeStamp);

} catch (UnsupportedEncodingException e) {

return strToSign;

}

return signWithHmac(strToSign, accessKeySecret);

}

private static String buildCanonicalString(String parameters, String httpMethod, String timeStamp) {

StringBuilder builder = new StringBuilder();

builder.append(httpMethod).append("&");

try {

builder.append(URLEncoder.encode("/", UTF\_8)).append("&");

if (null != timeStamp) {

builder.append(URLEncoder.encode(timeStamp, UTF\_8)).append("&");

}

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

if (null != parameters) {

String queryString = parameters;

// 去除#后面部分

int fragIdx = parameters.indexOf('#');

if (fragIdx >= 0) {

queryString = parameters.substring(0, fragIdx);

}

builder.append(buildqueryString(queryString));

}

return builder.toString();

}

private static String buildqueryString(String queryString) {

Map<String, String> params = getQueryStringMap(queryString);

StringBuilder builder = new StringBuilder("");

if (params.size() > 0) {

String[] names = params.keySet().toArray(new String[params.size()]);

Arrays.sort(names);

char separator='&';

for (int i=0; i<names.length; i++) {

String name = names[i];

if (0 != i) {

builder.append(separator);

}

builder.append(name);

String paramValue = params.get(name);

if (paramValue != null && paramValue.length() > 0) {

builder.append("=").append(paramValue);

}

}

}

String cononiStr = "";

try {

cononiStr = URLEncoder.encode(builder.toString(), UTF\_8);

} catch (Exception e) {

e.printStackTrace();

}

return cononiStr;

}

private static Map<String, String> getQueryStringMap(String queryString) {

Map<String, String> params = new TreeMap<String, String>();

if (null != queryString) {

String[] splits = queryString.split("&");

for (int i = 0; i< splits.length; i++) {

String query = splits[i];

String[] tmpSplit = query.split("=");

if (tmpSplit.length >= 2) {

String name = tmpSplit[0];

String value = tmpSplit[1];

params.put(name,value);

}

}

}

return params;

}

private static byte[] hmacsha256Signature(byte[] data, byte[] key) {

try {

SecretKeySpec signingKey = new SecretKeySpec(key, "HmacSHA256");

Mac mac = Mac.getInstance("HmacSHA256");

mac.init(signingKey);

return mac.doFinal(data);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

private static String signWithHmac(String strToSign, String accessKey) {

byte[] crypto = hmacsha256Signature(strToSign.getBytes(), accessKey.getBytes());

String signature = Base64.encodeBase64String(crypto).trim();

return signature;

}

}

1. （规范性）  
   公网认证示例

公网认证示例代码如下：

public static final String SIGN = "sign";

/\*\*

\* 生成MD5签名

\*

\* @param params 待签名参数

\* @param secret 签名密钥

\*/

public static String buildServerSign(Map<String, String> params, String secret) {

Map<String, String> map = new TreeMap<>(params);

StringBuilder sb = new StringBuilder();

map.forEach((key, value) -> {

if (StrUtil.isNotBlank(key) && !SIGN.equals(key)) {

sb.append(key).append(value);

}

});

String source = secret + sb.toString() + secret;

return DigestUtils.md5Hex(source).toUpperCase();

}