联友提供的接口 (大风车 -> DMS)

# APP创建订单

接口: webService 接口, 由联友提供接口定义

说明: 大风车APP下发的订单完整信息到联友DMS

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 订单编号 | orderNo | 必填 |  |
| 客户姓名 | custName | 必填 | 张三 |
| 微信ID | openId | 可空 |  |
| 手机 | phone | 必填 | 13312345678 |
| 身份证号 | certno | 可空 | 如果是公司可以选择不填写 |
| 车型18位码 | dmsModelCode | 必填 |  |
| 经销商编码 | dmsDealerCode | 必填 |  |
| 客户来源系统编码 | custSourceSystem | 固定值 | 固定为APP |
| 客户来源编码 | custSource | 必填 | 2 保有客户推荐  3 展厅自然来店  4 销售线索来店  9 开拓渠道  10 保有客户增购  11 保有客户置换  12 意向客户二次来店  13 大客户来店  14 二级网点客户来店  15 维修及其他目的来店客户 |
| 客户类型编码 | custType | 必填 | 0 个人  1 公司 |
| 证件类型编码 | certType | 可空 | 1身份证  如果是公司可以不填写传空 |
| 销售顾问ID | caUserId | 必填 |  |
| 来店日期 | comeTime | 必填, 默认为当天 |  |
| 建档日期 | createTime | 必填, 默认为当天 |  |
| 是否大客户 | isVip | 必填 | 1 是 0 否 |
| 大客户类型 | vipType | 当大客户为”是”时必填 | 1 租赁公司  2 出行平台  2A 单位个人购车  2B 企事业公务车  2C 政府公务车  3 企业  4 酒店  5 航空公司  6 股东及子公司/关联公司  7 总对总企业员工  8 留学生  9 公务员  10 VIP  11 媒体从业人员 |
| 性别 | sex | 可空 | 0 女 1男 |
| 省份ID | provinceCode | 可空 |  |
| 城市ID | cityCode | 可空 |  |
| 区县ID | regionCode | 可空 |  |
| 详细地址 | address | 必填 | 默认为无 |
| 公司名称 | compName | 当客户类型为”公司”时必填 |  |
| 公司性质代码 | compKind | 可空 | 1 国有企业  2 民营企业  3 三资企业  4 合资企业  5 外资企业  6 党政机关  7 事业单位  8 股份制企业 |
| 公司规模代码 | compOrgscale | 可空 | 1 50人以下  2 50人~100人  3 100人~200人  4 200~500人  5 500~1000人  6 1000~2000人  7 2000人以上 |
| 外观颜色编码 | carColor | 必填 色值枚举 |  |
| 内饰颜色编码 | carInColor | 必填 色值枚举 |  |
| 年款 | modelYear | 必填 |  |
| 购买形式编码 | buyModel | 必填 | 1. 首购  2. 换购  3. 增购 |
| 购车方式编码 | buyType | 必填 | 1. 全额车款 2. 按揭贷款 |
| 购车用途编码 | buyWay | 必填 | 1. 私用 2. 共用 3. 商用 4. 出租 |
| 销售渠道编码 | saleChannel | 必填 | 1. 散户零售 2. 政府采购 3. 企业采购 4. 4S所属集团内销   99. 其他 |
| 销售价格 | contractPrice | 可空, 数字, 单位元 |  |
| 应收订金 | deposit | 可空, 数字, 单位元 |  |
| 预计交车日期 | preDeliveryDate | 必填 yyyy-mm-dd |  |
| 下订时间 | orderTime | 必填  yyyy-mm-dd hh24:mi:ss |  |

返回:

# APP取消订单

接口: webService 接口, 由联友提供接口定义

说明: 大风车APP取消订单, 同步状态到联友

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 订单编号 | orderNo | 必填 |  |
| 申请员工ID | applyUserId | 必填 |  |
| 经销商编码 | dmsDealerCode | 必填 |  |
| 申请时间 | applyTime | 必填 |  |
| 退订原因 | reason | 必填, 已收全款时 |  |
| 退订说明 | remark | 必填, 已收全款时 |  |
| 审核员工ID | checkUserId | 必填 |  |
| 审核时间 | checkTime | 必填  yyyy-mm-dd hh24:mi:ss |  |

返回:

修改记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 修改日期 | 修改人 | 修改内容 |
| 1.0 | 2017/12/7 | 韩卓亮 | 初稿 |
| 1.1 | 2017/12/12 | 韩卓亮 | 线索接口增加字段, 增加渠道枚举字段下发接口 |
| 1.2 | 2017/12/14 | 韩卓亮 | 1. 将新增和改修合并成保存一个接口 2. 修改了经销商省市区字段 3. 补充了可空必填说明 4. 调整了下发渠道枚举参数结构 |
| 1.3 | 2017/12/14 | 韩卓亮 | 修改线索字段, 增加枚举说明和状态字段 |
| 1.4 | 2017/12/19 | 韩卓亮 | 1. 增加了C#调用例子  2. 同步用户增加是否是管理员字段 |
| 1.5 | 2017/12/20 | 韩卓亮 | 完善了联友接口参数部分 |
| 1.6 | 2017/12/27 | 韩卓亮 | 1. 保存线索接口, 增加年款, 车型颜色, 内饰颜色编码 非必填 2. 保存试驾车辆接口, 增加年款 必填, 车型颜色, 内饰颜色编码, 非必填 3. 增加保存车型颜色接口 4. 联友接口增加英文字段 |

大搜车提供的接口(DMS -> 大风车)

固定参数

/open/api.json

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 开放平台appKey | appKey | URL参数传递 | 大搜车提供 |
| 开放平台appSecret | appSecret | 加密生成appSign | 大搜车提供 |
| 开放平台appSign | appSign | URL参数传递 | 通过计算获得 |
| 时间戳 | timestamp | URL参数传递 | 当前时间, long型 |
| 接口名 | apiName | URL参数传递 | saveTestDriveRecord |

调用方式

第一步: 先根据所有参数进行字典序排序(不包括appSign),

第二步: 请求参数按照key1=value1&key2=value2拼接成字符串signString

第三步: 对signString进行base64加密signStringBase64

第四步: 将appSecret:signStringBase64进行sha1算法加密得到appSign

第五步: 生成请求URL

签名示例:

API saveTestDriveRecord

请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| appKey | appKey | URL参数传递 | yourAppKey |
| appSecret | appSecret | 加密生成appSign | yourAppSecret |
| appSign | appSign | URL参数传递 | 通过计算获得 |
| 时间戳 | timestamp | URL参数传递 | 当前时间, long型 |
| 接口名 | apiName | URL参数传递 | saveTestDriveRecord |
| 经销商编码 | dmsDealerCode | URL参数传递 | 000001 |
| 18位车型码 | dmsModelCode | URL参数传递 | 123456789012345678 |
| 车辆牌照 | plateNumber | URL参数传递 | 京A1234K |
| Vin码 | vinCode | URL参数传递 | 1G1BL52P7TR115520 |

示例代码(JAVA)

**package** com.souche;

**import** java.security.MessageDigest;

**import** java.security.NoSuchAlgorithmException;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.SortedMap;

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

**import** org.apache.commons.lang3.StringUtils;

**import** com.google.common.collect.Maps;

**public** **class** SoucheOpenTest {

**private** **final** String appKey = "yourAppKey";

**private** **final** String appSecret = "yourAppSecret";

**private** **final** String openApiHost = "http://open-retail.infiniti.com.cn/open/api.json";

**public** **static** **void** main(String[] args){

SoucheOpenTest test = **new** SoucheOpenTest();

Map<String,String> params = Maps.*newHashMap*();

params.put("dmsDealerCode", "000001");

params.put("dmsModelCode", "123456789012345678");

params.put("plateNumber", "京A1234K");

params.put("vinCode", "1G1BL52P7TR115520");

//第一步: 先根据所有参数进行字典序排序(不包括appSign)

SortedMap<String,String> sortedMap = test.sortMap("addTestDriveRecord",params);

//第二步: 请求参数按照key1=value1&key2=value2拼接成字符串signString

String signString = test.createSignString(sortedMap);

System.*out*.println("signString:"+signString);

//第三步: 对signString进行base64加密signStringBase64

//第四步: 将appSecret:signStringBase64进行sha1算法加密得到appSign

String sign = test.createSign(signString);

System.*out*.println("sign:"+sign);

//第五步: 生成请求URL

String requestUrl = test.createRequestUrl(signString, sign);

System.*out*.println(requestUrl);

}

/\*\*

\* 第一步: 先根据所有参数进行字典序排序(不包括appSign)

\* **@param** apiName

\* **@param** params

\* **@return**

\*/

**private** SortedMap<String,String> sortMap(String apiName, Map<String, String> params){

SortedMap<String,String> sortMap = Maps.*newTreeMap*();

Date now = **new** Date();

String timestamp = now.getTime()+"";

sortMap.put("timestamp", timestamp);

sortMap.put("appKey", appKey);

sortMap.put("apiName", apiName);

**if**(params == **null** || params.isEmpty()){

**return** sortMap;

}

**for**(String key : params.keySet()){

sortMap.put(key, params.get(key));

}

**return** sortMap;

}

/\*\*

\* 第二步: 请求参数按照key1=value1&key2=value2拼接成字符串signString

\* **@param** sortMap

\* **@return**

\*/

**private** String createSignString(Map<String,String> sortMap){

StringBuffer sb = **new** StringBuffer();

List<String> keys = **new** ArrayList<String>(sortMap.keySet());

Collections.*sort*(keys);

**for** (**int** i = 0; i < keys.size(); i++) {

String k = keys.get(i);

String v = sortMap.get(k);

**if**(StringUtils.*isNotEmpty*(v)) {

sb.append(k + "=" + v + "&");

}

}

String signString = sb.toString();

**if**(signString.endsWith("&") && signString.length() >= 2){

signString = signString.substring(0, signString.length() - 1);

}

**return** signString;

}

/\*\*

\* 第三步: 对signString进行base64加密signStringBase64

\* 第四步: 将appSecret:signStringBase64进行sha1算法加密得到appSign

\* **@param** signString

\* **@return**

\*/

**private** String createSign(String signString){

String signStringBase64 = base64(signString);

String appSign = appSecret+":"+ signStringBase64;

**return** sha1(appSign);

}

/\*\*

\* base64加密

\* **@param** s

\* **@return**

\*/

**private** String base64(String s){

**if**(StringUtils.*isEmpty*(s)){

**return** "";

}

s = **new** String(Base64.*encodeBase64*(s.getBytes()));

**return** s;

}

/\*\*

\* sha1加密

\* **@param** str

\* **@return**

\*/

**private** String sha1(String str){

MessageDigest md = **null**;

String tmpStr = "";

**try** {

md = MessageDigest.*getInstance*("SHA-1");

// 将三个参数字符串拼接成一个字符串进行sha1加密

**byte**[] digest = md.digest(str.getBytes());

tmpStr = byteToStr(digest);

} **catch** (NoSuchAlgorithmException e) {

e.printStackTrace();

}

**return** tmpStr;

}

/\*\*

\* 将字节数组转换为十六进制字符串

\*

\* **@param** byteArray

\* **@return**

\*/

**private** String byteToStr(**byte**[] byteArray) {

String strDigest = "";

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

strDigest += *byteToHexStr*(byteArray[i]);

}

**return** strDigest;

}

/\*\*

\* 将字节转换为十六进制字符串

\*

\* **@param** mByte

\* **@return**

\*/

**private** **static** String byteToHexStr(**byte** mByte) {

**char**[] Digit = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };

**char**[] tempArr = **new** **char**[2];

tempArr[0] = Digit[(mByte >>> 4) & 0X0F];

tempArr[1] = Digit[mByte & 0X0F];

String s = **new** String(tempArr);

**return** s;

}

/\*\*

\* 第五步: 生成请求URL

\* **@param** sign

\* **@param** signString

\* **@return**

\*/

**private** String createRequestUrl(String signString, String sign){

**return** openApiHost + "?" + signString + "&appSign=" + sign;

}

}

示例代码(C#)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Security.Cryptography;

namespace SoucheOpenTestApplication1

{

class SoucheOpenTest

{

private const string appKey = "yourAppKey";

private const string appSecret = "yourAppSecret";

private const string openApiHost = "http://open-retail.infiniti.com.cn/open/api.json";

public static void Main()

{

SoucheOpenTest test = new SoucheOpenTest();

Dictionary<string, string> param = new Dictionary<string, string>();

param.Add("dmsDealerCode", "000001");

param.Add("dmsModelCode", "123456789012345678");

param.Add("plateNumber", "京A1234K");

param.Add("vinCode", "1G1BL52P7TR115520");

//第一步： 先根据所有参数进行字典序排序（不包括appSign）

SortedDictionary<string, string> sortedMap = test.sortMap("addTestDriveRecord", param);

//第二步： 请求参数按照key1=value1&key2=value2拼接成字符串signString

string signString = test.createSignString(sortedMap);

Console.WriteLine("signString:" + signString);

//第三步： 对signString进行base64加密signStringBase64

//第四步： 将appSecret:signStringBase64进行sha1算法加密得到appSign

string sign = test.createSign(signString);

Console.WriteLine("sign:" + sign);

//第五步： 生成请求URL

string requestUrl = test.createRequestUrl(signString, sign);

Console.WriteLine(requestUrl);

}

/\*\*

\* 第一步：先根据所有参数进行字典排序

\*/

private SortedDictionary<string, string> sortMap(string apiName, Dictionary<string, string> param)

{

SortedDictionary<string, string> sortMap = new SortedDictionary<string, string>();

string timestamp = getCurrentTimestamp();

sortMap.Add("timestamp", timestamp);

sortMap.Add("appKey", appKey);

sortMap.Add("apiName", apiName);

if (param == null || param.Count == 0)

{

return sortMap;

}

foreach(string key in param.Keys)

{

sortMap.Add(key, param[key]);

}

return sortMap;

}

/\*\*

\* 第二步： 请求参数按照key1=value1&key2=value2拼接成字符串signString

\*/

private string createSignString(SortedDictionary<string, string> sortMap)

{

StringBuilder sb = new StringBuilder();

foreach (string key in sortMap.Keys)

{

string value = sortMap[key];

if (value != null && value.Length > 0)

{

sb.Append(key + "=" + value + "&");

}

}

string signString = sb.ToString();

if (signString.EndsWith("&") && signString.Length >= 2)

{

signString = signString.Substring(0, signString.Length - 1);

}

return signString;

}

/\*\*

\* 获取当前时间戳

\*/

private string getCurrentTimestamp()

{

DateTime now = DateTime.Now;

DateTime dtStart = TimeZone.CurrentTimeZone.ToLocalTime(new DateTime(1970, 1, 1));

TimeSpan toNow = now.Subtract(dtStart);

long timeStamp = toNow.Ticks;

timeStamp = long.Parse(timeStamp.ToString().Substring(0, timeStamp.ToString().Length - 4));

return timeStamp.ToString();

}

/\*\*

\* 第三步： 对signString进行base64加密signStringBase64

\* 第四步： 将appSecret:signStringBase64进行sha1算法加密得到appSign

\*/

private string createSign(string signString)

{

string signStringBase64 = base64(signString);

string appSign = appSecret + ":" + signStringBase64;

return sha1(appSign);

}

/\*\*

\* base64加密

\*/

private string base64(string s)

{

byte[] bytes = Encoding.UTF8.GetBytes(s);

return Convert.ToBase64String(bytes);

}

/\*\*

\* sha1加密

\*/

private string sha1(string s)

{

try

{

SHA1 sha1 = new SHA1CryptoServiceProvider();

byte[] bytes\_in = Encoding.UTF8.GetBytes(s);

byte[] bytes\_out = sha1.ComputeHash(bytes\_in);

string result = BitConverter.ToString(bytes\_out);

result = result.Replace("-", "");

return result;

}

catch (Exception ex)

{

throw new Exception("SHA1加密出错：" + ex.Message);

}

}

/\*\*

\* 第五步： 生成请求URL

\*/

private string createRequestUrl(string signString, string sign)

{

return openApiHost + "?" + signString + "&appSign=" + sign;

}

}

}

# 保存试乘试驾记录

接口: saveTestDriveRecord

频次: 实时调用, 无需批量同步

说明: 由DMS发起保存(根据vinCode新增或修改)一条试乘试驾记录

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 经销商编码 | dmsDealerCode | 必填, 经销在DMS的编码 |  |
| 车型18位码 | dmsModelCode | 必填 | 123456789012345678 |
| 车型年款 | modelYear | 必填 | 2017 |
| 车型颜色编码 | carColor | 可空 | GAG |
| 车型内饰编码 | carInColor | 可空 | G |
| 车辆牌照 | plateNumber | 必填 | 浙A1234K |
| VIN码 | vinCode | 必填 | 1G1BL52P7TR115520 |
| 行驶里程 | mileage | 可空, 单位公里 |  |
| 登记日期 | registerDate | 可空 |  |
| 启用状态 | status | 必填 | 0 – 停用 1 – 启用 |

返回:

{

“code”: “200”,

“data”: true,

“msg”: “success”,

“success”: true,

“traceId”: “xxxxxxx\_xxx”

}

code: 具体API错误信息编码, 成功状态保持为200

data: 具体是否添加成功, Boolean类型

msg: 错误信息, 成功的话显示success

success: true/false 是否调用成功

traceId: 请求链路ID, 用于内部定位问题

# 保存销售线索

接口: saveSalesClue

频次: 实时调用, 无需批量同步

说明: 由DMS发起保存(根据dmsClueId新增或修改)一条线索记录

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 经销商编码 | dmsDealerCode | 必填,经销在DMS的编码 |  |
| 线索ID | dmsClueId | 必填,线索在DMS的唯一编码 |  |
| 销售顾问ID | dmsUserId | 可空,销售顾问ID |  |
| 意向车系 | dmsSeriesCode | 必填 | 车系编码 |
| 车型18位码 | dmsModelCode | 可空 | 123456789012345678 |
| 车型年款 | modelYear | 可空 | 2017 |
| 车型颜色编码 | carColor | 可空 | GAG |
| 车型内饰编码 | carInColor | 可空 | G |
| 客户姓名 | name | 必填, 潜客信息 |  |
| 客户性别 | sex | 可空, 性别 | 0 – 男 1 – 女 |
| 客户手机 | phone | 必填, 潜客信息 |  |
| 来源一级 | platform1 | 必填, 渠道一级 | 一级枚举 |
| 来源二级 | platform2 | 必填, 渠道二级 | 二级枚举 |
| 来源三级 | platform3 | 必填, 渠道三级 | 三级枚举 |
| 客户等级 | level | 可空, 意向级别 | H 计划7天内买车  A 计划一个月内买车  B 计划三个月内买车  C 计划三个月后买车  E 计划6-12个月买车  L 无效等级 |
| 留资时间 | createTime | 可空, 创建时间 | 如不填则按当前时间 |
| 预约到店时间 | arriveStoreTime | 可空 |  |
| 计划回访时间 | planCallbackTime | 可空, 计划回访时间 |  |
| 购车预算 | budget | 可空, 枚举, 缺省为0 | 0 未知   1. 20万以下 2. 20-30万 3. 30-40万 4. 40-50万 5. 50-60万 6. 60-70万 7. 70-80万 8. 80万以上 |
| 状态 | status | 必填, 线索状态 | 0 – 有效 1 – 无效 |
| 备注 | comment | 可空, 接触方式 |  |
| 跟进时间 | followTime | 可空,跟进时间 | yyyy-MM-dd HH:mm:ss |
| 此线索的描述、车辆信息等 | description | 可空 | 线索的描述内容 |
| 预算在购买范围内 | isBudgetRange | 可空，单选 | 1 是 0 否 不填传空 |
| 竞品车系 | competeCode | 可空，单选，有传车系code，没有传空 | 大搜车车系编码 |
| 计划购车时间 | planBuyCarTime | 可空，单选，计划买车的时间 | ONE\_WEEK 1周内 ONE\_MONTH 1个月内 THREE\_MONTH 3个月内 SIX\_MONTH 6个月内 ONE\_YEAR 1年内 ONE\_YEAR\_AFTER 1年后 不填传空值 |
| 线下体验过 | isObserved | 可空，单选 | 1 是 0 否 不填传空 |
| 在可售区域 | isSellRegion | 可空，单选 | 1 是 0 否 不填传空 |
| 客户画像url | customerImageUrl | 可空 |  |

返回:

{

“code”: “200”,

“data”: true,

“msg”: “success”,

“success”: true,

“traceId”: “xxxxxxx\_xxx”

}

code: 具体API错误信息编码, 成功状态保持为200

data: 具体是否添加成功, Boolean类型

msg: 错误信息, 成功的话显示success

success: true/false 是否调用成功

traceId: 请求链路ID, 用于内部定位问题

# 下发销售线索渠道枚举

接口: updateSalesCluePlatfrom

频次: 实时调用, 无需批量同步

说明: 由DMS发起下发销售线索渠道枚举, 全量更新

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 渠道枚举 | platforms | 必填json格式, 将全量数据通过POST方式提交大搜车全量更新 | 数组格式 |

具体格式:

[{

id: ‘1’,

code: ‘webPlatform’,

name: ‘网络平台’,

parentCode: ‘’,

level: 1,

dmsDealerCode:’’   (此为新加字段，标识经销商编号，如果为DFI来源则传空字符串)

},{

id:’2’,

code: ‘che168’,

name: ‘汽车之家’,

parentCode: ‘webPlatform’,

level: 2,

dmsDealerCode:’’

},{

id:’3’,

code: ‘che168-myCar’,

name: ‘我的爱车’,

parentCode: ‘che168’,

level: 3,

dmsDealerCode:’I4205I’

},{

…

}]

返回:

{

“code”: “200”,

“data”: true,

“msg”: “success”,

“success”: true,

“traceId”: “xxxxxxx\_xxx”

}

code: 具体API错误信息编码, 成功状态保持为200

data: 具体是否添加成功, Boolean类型

msg: 错误信息, 成功的话显示success

success: true/false 是否调用成功

traceId: 请求链路ID, 用于内部定位问题

# 提醒更新客户画像链接内容

接口: callUpdatedCustomerImage

频次: 实时调用

说明: 由DMS发起

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 经销商编码 | dmsDealerCode | 必填,经销商在DMS的编码 |  |
| 客户手机 | phone | 必填，客户手机号 |  |
| 客户画像更新时间 | updateTime | 选填，如果汽车之家画像更新了，但是没有时间可以不填 |  |
| 大类编码（一级渠道） | firstChannel | 选填 |  |
| 中类编码（二级渠道） | secondChannel | 选填 |  |
| 小类编码（三级渠道） | thirdChannel | 选填 |  |

返回:

{

“code”: “200”,

“data”: true,

“msg”: “success”,

“success”: true,

“traceId”: “xxxxxxx\_xxx”

}

code: 具体API错误信息编码, 成功状态保持为200

data: 具体是否告知销售成功，Boolean类型

msg: 错误信息, 成功的话显示success

success: true/false 是否调用成功

traceId: 请求链路ID, 用于内部定位问题

# 保存经销商

接口: saveShop

频次: 实时调用, 无需批量同步

说明: 由DMS发起保存(根据dmsDealerCode新增或修改)一个经销商记录

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 经销商编码 | dmsDealerCode | 必填, 经销在DMS的编码 |  |
| 全称 | name | 必填 |  |
| 简称 | shortName | 必填 |  |
| 省名称 | provinceName | 可空, 省名称 |  |
| 市名称 | cityName | 可空, 市名称 |  |
| 区名称 | regionName | 可空, 区名称 |  |
| 省编码 | provinceCode | 必填, 省编码 |  |
| 市编码 | cityCode | 必填, 市编码 |  |
| 区编码 | regionCode | 必填, 区编码 |  |
| 地址 | address | 必填, 地址 |  |
| 服务电话 | phone | 可空, 销售热线 |  |
| 状态 | status | 必填, 在网、退网(启用状态) | 0 – 在网 1 – 退网 |

返回:

{

“code”: “200”,

“data”: true,

“msg”: “success”,

“success”: true,

“traceId”: “xxxxxxx\_xxx”

}

code: 具体API错误信息编码, 成功状态保持为200

data: 具体是否添加成功, Boolean类型

msg: 错误信息, 成功的话显示success

success: true/false 是否调用成功

traceId: 请求链路ID, 用于内部定位问题

# 保存账号信息

接口: saveUserAccount

频次: 实时调用, 无需批量同步

说明: 由DMS发起保存(根据dmsUserId新增或修改)一个员工账号, 密码由大风车默认生成

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 经销商编码(归属) | dmsDealerCode | 必填, 经销在DMS的编码, 包括总部 |  |
| 账号ID | dmsUserId | 必填, 账号在DMS中的编码(员工编码) |  |
| 姓名 | name | 必填, 姓名 |  |
| 性别 | sex | 可空, 性别 | 0 – 男 1 – 女 |
| 手机号 | phone | 必填, 登录手机号 |  |
| 登录账号 | account | 可空, 默认手机号 |  |
| 状态 | status | 可空, 默认在职  在职、离职、停用 | 0 – 在职 1 – 离职 2 – 在职 |
| 是否是管理员 | isAdmin | 是否是管理员 | true/false |
| 创建时间 | createTime | 可空, 如果为空用系统当前时间 |  |
| 离职时间 | resignTime | 可空 |  |

返回:

{

“code”: “200”,

“data”: true,

“msg”: “success”,

“success”: true,

“traceId”: “xxxxxxx\_xxx”

}

code: 具体API错误信息编码, 成功状态保持为200

data: 具体是否添加成功, Boolean类型

msg: 错误信息, 成功的话显示success

success: true/false 是否调用成功

traceId: 请求链路ID, 用于内部定位问题

# 保存车辆颜色信息

接口: saveCarColor

频次: 实时调用, 无需批量同步

说明: 由DMS发起保存(根据dmsModelCode新增或修改)一个车型的颜色

参数:

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 字段 | 说明 | 示例 |
| 车型18位码 | dmsModelCode | 必填 |  |
| 颜色编码 | carColor | 车型颜色编码 | #ECECEC |
| 颜色名称 | carColorName | 车型颜色名称 | 奶奶灰 |
| 内饰编码 | carInColor | 车型内饰颜色编码 | #000000 |
| 内饰名称 | carInColorName | 车型内饰颜色 | 黑色真皮 |
| 年款 | modelYear | 车型年款 | 2014 |
| 状态 | status | 启用/停用 | 0 – 启用 1 – 停用 |

返回:

{

“code”: “200”,

“data”: true,

“msg”: “success”,

“success”: true,

“traceId”: “xxxxxxx\_xxx”

}

code: 具体API错误信息编码, 成功状态保持为200

data: 具体是否添加成功, Boolean类型

msg: 错误信息, 成功的话显示success

success: true/false 是否调用成功

traceId: 请求链路ID, 用于内部定位问题

# 数据字典

关于省市区数据字典, 由联友提供一份导出的数据, 由大搜车负责映射完, 发送回给联友