**Data push service access manual**

**Version： V1.1**

**Date：March 29, 2018**

Document revision record

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Standard revision history record | | | | | | | | |
| Version | Date | | Reviser | Revision page | | Revise the content | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
|  |  | |  |  | |  | | |
| 会 签 | | | | | | | | |
| Management body | | Signed advice | | | | Signature | | Date |
|  | |  | | | |  | |  |
|  | |  | | | |  | |  |
|  | |  | | | |  | |  |
|  | |  | | | |  | |  |
|  | |  | | | |  | |  |
|  | |  | | | |  | |  |
| Create | |  | | | Review | |  | |
| Audit | |  | | | Approval | |  | |

**Directory**

[Data push service access manual 2](#_Toc519789844)

[1 Introduction 6](#_Toc519789845)

[1.1 Executive summary 6](#_Toc519789846)

[1.2 Purpose of writing 6](#_Toc519789847)

[1.3 Special token format description 6](#_Toc519789848)

[2 Public show 6](#_Toc519789849)

[2.1 Basic introduction 6](#_Toc519789850)

[2.2 Call process 6](#_Toc519789851)

[2.3 Delivery address 7](#_Toc519789852)

[2.4 Signature authentication 7](#_Toc519789853)

[2.4.1 instructions 7](#_Toc519789854)

[2.4.2 Parameter introduction 7](#_Toc519789855)

[2.4.3 Algorithm 7](#_Toc519789856)

[2.5 International processing 8](#_Toc519789857)

[2.6 Data type qualification 8](#_Toc519789858)

[2.6.1 Field type description 8](#_Toc519789859)

[2.6.2 Description of null values 8](#_Toc519789860)

[3 Data push service access process description 10](#_Toc519789861)

[4 The appendix 11](#_Toc519789862)

[4.1 Common error code 11](#_Toc519789863)

[4.2 Example signature algorithm 12](#_Toc519789864)

1. Introduction
   1. Executive summary

As the basis for the development of subscription data push services for third-party cloud platforms.

* 1. Purpose of writing

Provide basic and guidance for the development of Haier equipment data for third-party cloud platforms.

* 1. Special token format description

This document does not use other special document marks or formats.

The interface examples in this article all use sample data that is not real data.

1. Public show
   1. Basic introduction

A third party can subscribe to the information function of haier's equipment through data push service to realize the communication between haier's equipment data and the third party platform.

* 1. Call process

To use the interfaces in the documentation for third-party platforms, the following process is required.

Third party cloud platforms accept the messaging service path

Creating cloud applications

Registered

Enterprise developer

Subscribe to the message push service

Subscription message device type information

Device push interface calls

* 1. Delivery address

All interfaces provided in this document support HTTP and HTTPS protocol requests.

The interaction interface with data push is unified into REST interface based on HTTP or HTTPS. Utf-8 coding;When developing joint debugging, the application developer should provide the push address of the developer environment to receive push data; when going online, the application developer should provide the push address of the production environment to receive push data.

* 1. Signature authentication
     1. instructions

The data push platform server needs to sign the sending request, and the signature value of the signature calculation needs to be assigned to the sign attribute in the Header header (see the public part description) for the caller to perform signature verification.

* + 1. Parameter introduction

**The string to be signed is:：**SystemID+SystemKey +timestamp；

**SystemID：**Cloud application ID,within 40 characters,Haier U+ cloud platform globally unique;

**SystemKey：**The SystemKey applied to the application on the Haigeek network cannot be sent in plain text;

**timestamp：**Unix Timestamp（*YYMMDDhhmmss*）；

* + 1. Algorithm

The signature algorithm is to calculate the 32-bit lowercase SHA-256 value for the signature string. See the appendix for an example of the algorithm.

* 1. International processing

The retCode and retInfo returned by the interface response are not internationalized and are processed by the interface caller.

The internationalization of the interface involving business data is defined by passing the language parameter in the header. The specific international language code is shown in the appendix.

* 1. Data type qualification
     1. Field type description

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualified type** | **Instructions** | **Format** | **Json Sample** |
| DateTime | Date time type string | yyyy-MM-ddhh:mm:ss | {“lgTime”:“2013-10-08 08:00:00”} |
| Date | Date type string | yyyy-MM-dd | {“lgDate”:“2013-10-08”} |
| String | String |  | {“address”:“street 123”} |
| int | Int |  | {“age”:1234} |
| long | Long int |  | {“oid”:1234567890123} |
| double | Double |  | {“price”:12.35} |
| boolean | True or false |  | {“idOld”:true}。 |

* + 1. Description of null values

To avoid parsing errors, the return parameters of each interface of uws do not return a null value.

Required parameters, whether input or output, must have a value and cannot be null.

Non-required parameters are as follows:

Numerical type data（int、long、double）Only numbers are returned, including positive numbers, zeros, and negative numbers.

Boolean type data（boolean）Only return true or false

The above basic types do not themselves contain null values.

DateTime, Date, String, and structure type data. If it is null, the corresponding attribute will not be returned.

E.g:

***DateTime type***

birthday is a DateTime type, not null:

{"name":"Tom","age":23,"birthday":"2013-10-08 08:00:00","address":{"city":"beijing","street":"haidian"} }

When birthday is null, the birthday attribute does not return:

{"name":"Tom","age":23, "address":{"city":"beijing","street":"haidian"} }

***String type***

Name is a String type，not null：

{"name":"Tom","age":23,"birthday":"2013-10-08 08:00:00","address":{"city":"beijing","street":"haidian"} }

When name is null, the name attribute is not returned:

{"age":23,"birthday":"2013-10-08 08:00:00","address":{"city":"beijing","street":"haidian"} }

***Structure type***

address is a struct type, not null:

{"name":"Tom","age":23,"birthday":"2013-10-08 08:00:00","address":{"city":"beijing","street":"haidian"} }

When address is null, the address attribute does not return:

{"name":"Tom","age":23,"birthday":"2013-10-08 08:00:00" }

1. Data push service access process description

Description: The access process is an access process that combines cloud applications on the Haigeek network. Before the cloud application is not online, if you use the data push platform, you can only subscribe to the data push service offline.

The following information is required:

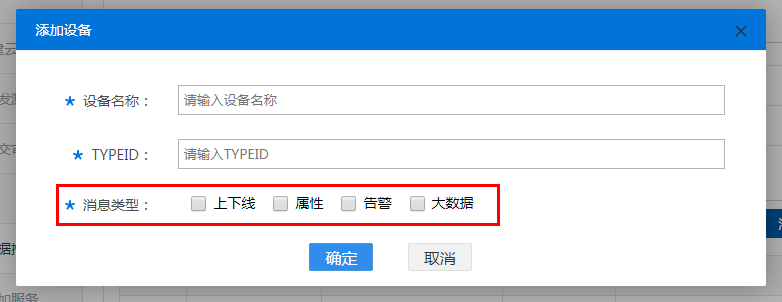
1、Provide SystemID and SystemKey



2、Provide device information (name, model), tyepid, and message type



3、Provides four REST interfaces for receiving data push platforms, online and offline paths, status paths, alarm paths, and big data paths. Currently, support for push based on typeid is supported.



1. The appendix
   1. Common error code

|  |  |
| --- | --- |
| **Error code** | **Info** |
| A00001 | Service unavailable |
| A00002 | Network exception |
| A00003 | Access or operation timeout |
| A00004 | Internal system error |
| A00005 | Database access exception |
| A00006 | Unknown exception |
| A00007 | Mail service exception |
| A00008 | Mail delivery failed |
| A00009 | The number of emails sent exceeded |
| B00001 | Missing required parameters |
| B00002 | Parameter type error |
| B00003 | Parameter values are out of range or not enumerated |
| B00004 | The parameters do not meet the rule requirements |
| B00006 | Parameter length error |
| B00007 | The parameters do not match the interface definition |
| C00001 | AppId and appKey validation failed |
| C00002 | AppServer has no access authorization |
| C00003 | Insufficient access |
| C00004 | Insufficient operation permission |
| C00005 | Repeated requests |
| C00006 | Unknown device type |
| C00007 | The appId configuration information is empty |
| C00008 | The appKey is empty |
| D00001 | Sign signature error |
| D00003 | Token does not exist and is not authenticated by Token |
| D00004 | Token has expired and is not authenticated by Token. |
| D00005 | Token is not created from this application and is not authenticated by Token. |
| D00006 | Session invalidation |
| D00007 | Not internal users |
| D00008 | User illegal |

* 1. Example signature algorithm

String getSign(String appId,StringappKey, String timestamp) throws Exception {

appId = appId.trim();

appKey = appKey.trim();

appKey = appKey.replaceAll("\"", "");

StringBuffersb = new StringBuffer();

sb.append(appId).append(appKey).append(timestamp);

MessageDigest md = null;

byte[] bytes = null;

try {

md = MessageDigest.getInstance("SHA-256");

bytes = md.digest(sb.toString().getBytes("utf-8"));

} catch (Exception e) {

e.printStackTrace();

}

returnBinaryToHexString(bytes);

}

String BinaryToHexString(byte[] bytes) {

StringBuilder hex = new StringBuilder();

String hexStr = "0123456789abcdef";

for (inti = 0; i<bytes.length; i++) {

hex.append(String.valueOf(hexStr.charAt((bytes[i] & 0xF0) >> 4)));

hex.append(String.valueOf(hexStr.charAt(bytes[i] & 0x0F)));

}

returnhex.toString();

}