参数签名教程 v1.0

版本	日期	更新
1.0	2022-9-26	新建

如何筛选出参与签名的参数

> ❗ 注意：可选参数的值为空或空字符串则需要将其排除签名，否则必须参与签名

> 以下列举了三种请求体类型传参方式筛选参与签名参数的规则

application/x-www-form-urlencoded

- 排除接口中定义为非必填且请求中未设置的参数
- 排除接口中定义为非必填且请求中设置为空串（""）的参数
- 排除 `sign` 参数

```
# 示例1
curl -X POST "https://dcapi.shunwang.com/cooperation/v1/messages/feishu/text/send" \
-H 'Content-Type:application/x-www-form-urlencoded; charset=utf-8'
-d 'app_id=aaa&timestamp=ttt&users=uuu&content=ccc&sign=sss'

# [参与] app_id=aaa, timestamp=ttt, users=uuu, content=ccc
# [排除] urgent 参数非必填且请求未设置参数
# [排除] sign 参数不参与签名

########################################

# 示例2
curl -X POST "https://dcapi.shunwang.com/cooperation/v1/messages/feishu/text/send" \
-H 'Content-Type:application/x-www-form-urlencoded; charset=utf-8'
-d 'app_id=aaa&timestamp=ttt&users=uuu&content=ccc&urgent=&sign=sss'

```

```
# [参与] app_id=aaa, timestamp=ttt, users=uuu, content=ccc
# [排除] urgent 参数非必填但请求设置了空串
# [排除] sign 参数不参与签名

#######################################

# 示例3
curl -X POST "https://dcapi.shunwang.com/cooperation/v1/messages/feishu/text/send" \
-H 'Content-Type:application/x-www-form-urlencoded; charset=utf-8'
-d 'app_id=aaa&timestamp=ttt&users=uuu&content=ccc&urgent=null&sign=sss'

# [参与] app_id=aaa, timestamp=ttt, users=uuu, content=ccc, urgent=null。请求中 urgent 并非空值，而是被视为字符 "null"，所以必须参与签名
# [排除] sign 参数不参与签名

#######################################

# 示例4
curl -X POST "https://dcapi.shunwang.com/cooperation/v1/messages/feishu/text/send" \
-H 'Content-Type:application/x-www-form-urlencoded; charset=utf-8'
-d 'app_id=aaa&timestamp=ttt&users=uuu&content=ccc&urgent=true&sign=sss'

# [参与] app_id=aaa, timestamp=ttt, users=uuu, content=ccc, urgent=true
# [排除] sign 参数不参与签名
```

multipart/form-data

- 排除接口中定义为非必填且请求中未设置的参数
- 排除接口中定义为非必填且请求中设置为空串（""）的参数
- 排除值为字节流类型的参数
- 排除 `sign` 参数

```
# 示例1
curl -X POST 'https://dcapi.shunwang.com/cooperation/v1/resources/feishu/image/put' \
-F 'image=@"/xxx/xxx.png";filename="xxx.png"' \
-F 'image_type=message' \
-F 'app_id=aaa' \
-F 'timestamp=ttt' \
-F 'sign=sss'

```

```
# 示例1中参与签名的参数：app_id=aaa, timestamp=ttt, image_type=message
# [排除] image 参数为字节流类型
# [排除] use_to, image_name 参数非必填且请求未设置
# [排除] sign 参数不参与签名

########################################

# 示例2
curl -X POST
'https://dcapi.shunwang.com/cooperation/v1/resources/feishu/image/put' \
-F 'image=@"/xxx/xxx.png";filename="xxx.png"' \
-F 'image_type=message' \
-F 'app_id=aaa' \
-F 'timestamp=ttt' \
-F 'sign=sss' \
-F 'use_to=' \
-F 'image_name=changed.png'

# [参与] app_id=aaa, timestamp=ttt, image_type=message, image_name=changed.png
# [排除] image 参数为字节流类型
# [排除] use_to, 参数非必填但请求设置为空串
# [排除] sign 参数不参与签名
```

application/json

- 排除请求体中JSON对象
- 排除接口中定义为非必填且请求中未设置的参数
- 排除接口中定义为非必填且请求中设置为空串（""）的参数
- 排除 `sign` 参数

```
# 示例1
curl -X POST 'https://dcapi.shunwang.com/digital-human/v1/ai/intention/parse?
app_id=aaa&timestamp=ttt&sign=sss' \
-H 'Content-Type:application/json; charset=utf-8' \
-d '{"query":"跳舞","scene":[],"history_intentions":[]}'

# [参与] app_id=aaa, timestamp=ttt
# [排除] JSON请求
# [排除] sign 参数不参与签名

########################################

# 示例2
```

```
# 注: type, user_id 参数只是为了此处演示效果而添加的, 实际接口中不存在这二参数
curl -X POST 'https://dcapi.shunwang.com/digital-human/v1/ai/intention/parse?app_id=aaa&timestamp=ttt&sign=sss&type=&user_id=1' \
-H 'Content-Type:application/json; charset=utf-8' \
-d '{"query":"跳舞","scene":[],"history_intentions":[]}'

# [参与] app_id=aaa, timestamp=ttt, user_id=1
# [排除] JSON请求体
# [排除] type 参数非必填但设置了空串
# [排除] sign 参数不参与签名
```

签名规则

> 为了便于说明签名规则，以下假定了参与签名的参数

> ❗ **注意：以下参与签名的参数仅为了便于说明规则而模拟设置，不能用于真实接口**

```
{
    "app_id": "PQUNIRPjFa8iDUlcVwtAJue6ODAOXp1a", // 注:真实数据来源于数智中心分配
    "timestamp": "20190101010101", // 注:真实数据应为当时时间戳

    // 以下参数在真实调用时应以对应接口文档提供的为准
    "user_name": "张三",
    "user_id": 123456
}

// 密钥: X5jbMENw2idWS3wcAnDyAylCpU53gYdK
```

1. 参与签名的参数名称按字母升序排序

示例：app_id, timestamp, user_id, user_name

2. 按排序后的参数顺序组织签名数据. 每个参数值之间以英文"|"连接, 最后需要加上 "签名密钥"

> ❗ **注意：签名密钥** 必须要加且必须要在末尾

示例：PQUNIRPjFa8iDUlcVwtAJue6ODAOXp1a|20190101010101|123456|张三|X5jbMENw2idWS3wcAnDyAylCpU53gYdK

3. 对转义后的签名数据进行URLEncode编码

示例：
PQUNIRPjFa8iDUlcVwtAJue6ODAOXp1a%7C20190101010101%7C123456%7C%E5%BC%A0%E4%B8%89%7CX5jbMENw2idWS3wcAnDyAylCpU53gYdK

4. 对签名数据进行特殊字符转义

> ❗ 注意：在URLEncode后不同语言之间还会存在略微差异，需要针对性进行替换

4.1 Java

> 不需要进行特殊字符转义

4.2 Go

替换前	替换后
~	%7E
%2A	*

4.3 Python

4.4 C++

4.5 C#

5. 对签名数据进行MD5（HEX小写）编码, 生成最终的签名数据sign

示例：27b5f95cd990bb2deb5066fc302dc9a3

签名示例

> ❗ 以下示例的目的是为了说明签名的思路，代码有些冗余，在理解后可以自行重构

Java

> 示例需要依赖 **guava**, **lombok** 库

```java
package signature;

import com.google.common.base.Charsets;
import com.google.common.base.Preconditions;
import com.google.common.base.Strings;
import com.google.common.collect.Maps;
import lombok.extern.slf4j.Slf4j;

import java.net.URLEncoder;
import java.security.MessageDigest;
import java.util.Objects;
import java.util.Optional;
import java.util.TreeMap;
import java.util.stream.Collectors;

/**
 * Java版参数签名示例
 */
@Slf4j
public class Signatures {

    private static final char[] HEX_CHARS = new char[] {
        '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e',
    };

    public static void main(String[] args) {
        TreeMap<String, String> params = Maps.newTreeMap();
        params.put("app_id", "PQUNIRPjFa8iDUlcVwtAJue6ODAOXp1a");
        params.put("timestamp", "20190101010101");
        params.put("user_name", "张三");
        params.put("user_id", "123456");

        sign("X5jbMENw2idWS3wcAnDyAylCpU53gYdK", params);
    }

    /**
     * 参数签名
     *
     * @param signKey 签名密钥
     * @param params  参与签名的参数
     */
    public static String sign(String signKey, TreeMap<String, String> params) {
        Preconditions.checkState(StringUtils.isNotEmpty(signKey), "未指定签名MD5Key
        Preconditions.checkState(params != null && !params.isEmpty(), "未指定需要签

        log.info("步骤1-参数名称排序: {}", params.keySet());

```

```
48          return Optional
49              .of(concat(signKey, params))
50              .flatMap(Signatures::urlEncode)
51              .flatMap(Signatures::md5)
52              .orElseThrow(() -> new RuntimeException("参数签名失败"));
53      }
54
55      /**
56       * 组织签名数据
57       *
58       * @param signKey 签名密钥
59       * @param params  参与签名的参数
60       * @return 通过"|"合并后并加上密钥的签名数据
61       */
62      private static String concat(String signKey, TreeMap<String, String> params)
63          String sign = params
64              .entrySet()
65              .stream()
66              // 排除 sign 参数及值为空（null）或空串（""）的参数
67              .filter(entry -> !Objects.equals(entry.getKey(), "sign") && !Strings.is
68              .map(entry -> entry.getValue().trim())
69              .collect(Collectors.joining("|"))
70              .concat("|")
71              .concat(signKey);
72
73          log.info("步骤2-组织签名数据: {}", sign);
74
75          return sign;
76      }
77
78      /**
79       * 对签名数据进行URLEncode编码
80       *
81       * @param data 签名数据
82       * @return URLEncode编码后的签名数据
83       */
84      private static Optional<String> urlEncode(String data) {
85          try {
86              String urlEncode = URLEncoder.encode(data, Charsets.UTF_8.name());
87              log.info("步骤4-URLEncode: {}", urlEncode);
88
89              return Optional.of(urlEncode);
90
91          } catch (Exception e) {
92              return Optional.empty();
93          }
94      }
```

```java

    /**
     * 对签名数据进行MD5编码
     *
     * @param data URLEncode编码后的签名数据
     * @return MD5编码后的签名数据
     */
    private static Optional<String> md5(String data) {
        try {
            MessageDigest messageDigest = MessageDigest.getInstance("MD5");
            messageDigest.update(data.getBytes(Charsets.UTF_8));

            return hex(messageDigest.digest());
        } catch (Exception e) {
            return Optional.empty();
        }
    }

    /**
     * 对签名数据进行HEX编码
     *
     * @param data MD5编码后的签名数据
     * @return HEX编码后的签名数据
     */
    private static Optional<String> hex(byte[] data) {
        char[] buf = new char[data.length * 2];
        int i = 0;

        for (int var3 = 0; i < data.length; ++i) {
            buf[var3++] = HEX_CHARS[data[i] >>> 4 & 15];
            buf[var3++] = HEX_CHARS[data[i] & 15];
        }

        String hex = new String(buf);
        log.info("步骤5-MD5: {}", hex);

        return Optional.of(hex);
    }

}
```

Go

```go
package signature

```

```go
import (
    "crypto/md5"
    "encoding/hex"
    "fmt"
    "net/url"
    "sort"
    "strings"

    log "github.com/sirupsen/logrus"
)

//
// Sign - 针对传入的接口参数进行签名
//
// Parameters:
//      signKey - 签名密钥
//      parameters - 需要签名的参数集
//
// Returns:
//      string - 参数签名
//
func Sign(signKey string, parameters map[string]string) string {
    // 1. 将参数名称按字母升序排序
    sortedNames := doSortNames(parameters)
    log.Tracef("sorted names: %s", sortedNames)

    // 2. 将每个参数用英文"|"分隔,并以提供的接口密钥结尾
    data := doJoinValues(sortedNames, parameters) + "|" + signKey
    log.Tracef("data: %s", data)

    // 3. 将签名数据URLEncode编码
    encodeData := url.QueryEscape(data)
    log.Tracef("encode data: %s", encodeData)

    // 4. 对特殊字符转义
    escapeData:= doEscape(encodeData)
    log.Tracef("escape data: %s", escapeData)

    // 4. 将签名数据使用MD5加密(加密时需要UTF-8编码)
    md5Data := doMd5(escapeData)
    log.Tracef("md5 data: %s", md5Data)

    // 5. 针对中台接口的一些特殊字符进行处理
    // doEscape(md5Data)

    return md5Data
}
```

```go

//
// doSortNames - 将参数集的名称按升序排序
//
// Parameters:
//      parameters - 需要签名的参数集
//
// Returns:
//      []string - 按升序排序后的参数名集
//
func doSortNames(parameters map[string]string) []string {
    var sortedNames []string
    for name, value := range parameters {
        // 排除 sign 参数
        // 排除值为 nil 或 空串 的参数
        if name == "sign" || len(value) == 0 || len(value) > 6 && strings.Compar
            continue
        }
        sortedNames = append(sortedNames, name)
    }

    sort.Strings(sortedNames)

    return sortedNames
}

//
// doJoinValues - 按排序后的参数名对应的参数值以"|"进行连接
//
// Parameters:
//      sortedNames - 按升序排序后的参数名集
//      parameters - 需要签名的参数集
//
// Returns:
//      string - 以"|"进行连接的参数值
//
func doJoinValues(sortedNames []string, parameters map[string]string) string {
    var values []string

    for _, name := range sortedNames {
        values = append(
            values,
            fmt.Sprintf("%v", parameters[name]),
        )
    }

    return strings.Join(values, "|")
```

```go
}

//
// doEncode - 将指定数据进行UrlEncode加密
//
// Parameters:
//      data - 需要进行UrlEncode的数据
//
// Returns:
//      string - 经过UrlEncode加密后的数据
//
func doEncode(data string) string {
    return doEscape(url.QueryEscape(data))
}

//
// doMd5 - 将指定数据进行MD5加密
//
// Parameters:
//      data - 需要进行MD5的数据
//
// Returns:
//      string - 经过MD5加密后的数据
//
func doMd5(data string) string {
    md5Encryptor := md5.New()
    md5Encryptor.Write([]byte(data))

    return hex.EncodeToString(md5Encryptor.Sum(nil))
}

//
// doEscape - 特殊字符转义
//
// Parameters:
//      data - 需要转义的字符串
//
// Returns:
//      string - 转义后的字符串
//
func doEscape(data string) string {
    data = strings.ReplaceAll(data, "~", "%7E")
    return strings.ReplaceAll(data, "%2A", "*")
}

```

Python

C++

C#