本文档为通过Open Star 平台添加终端设备应用时选择Webhook协议进行消息获取时，需要使用的第三方自行编写webservice进行调用，下列给出go、nodejs语言调用示例代码，仅供参考，如需其他方法请参照开放API进行编写调用

Go语言方法示例代码如下：

package main

import (

"fmt"

"net/http"

// "strings"

"log"

"io/ioutil"

"flag"

"bytes"

    "crypto/aes"

    "crypto/cipher"

    "encoding/base64"

    //"encoding/hex"

)

func AesEncrypt(plantText, key []byte) ([]byte, error) {

    block, err := aes.NewCipher(key) //选择加密算法

    if err != nil {

        return nil, err

    }

    if block.BlockSize() != len(key) {

        return nil, fmt.Errorf("key length must equal block size")

    }

    plantText = PKCS7Padding(plantText, block.BlockSize())

    blockModel := cipher.NewCBCEncrypter(block, key)

    ciphertext := make([]byte, len(plantText))

blockModel.CryptBlocks(ciphertext, plantText)

    return ciphertext, nil

}

func PKCS7Padding(ciphertext []byte, blockSize int) []byte {

    padding := blockSize - len(ciphertext)%blockSize

    padtext := bytes.Repeat([]byte{byte(padding)}, padding)

    return append(ciphertext, padtext...)

}

func AesDecrypt(ciphertext, key []byte) ([]byte, error) {

    keyBytes := []byte(key)

    block, err := aes.NewCipher(keyBytes) //选择加密算法

    if err != nil {

        return nil, err

    }

    if block.BlockSize() != len(key) {

        return nil, fmt.Errorf("key length must equal block size")

    }

    blockModel := cipher.NewCBCDecrypter(block, keyBytes)

    plantText := make([]byte, len(ciphertext))

    blockModel.CryptBlocks(plantText, ciphertext)

    plantText = PKCS7UnPadding(plantText, block.BlockSize())

    return plantText, nil

}

func PKCS7UnPadding(plantText []byte, blockSize int) []byte {

    length := len(plantText)

    unpadding := int(plantText[length-1])

    return plantText[:(length - unpadding)]

}

func sayhelloName(w http.ResponseWriter, r \*http.Request) {

r.ParseForm()

s, \_ := ioutil.ReadAll(r.Body)

fmt.Println("body: ",string(s))

decodeBytes,\_ := base64.StdEncoding.DecodeString(string(s))

fmt.Println("decodeBytes: ",string(decodeBytes))

key:=[]byte(\*key\_t)

decodeAES,\_:= AesDecrypt(decodeBytes,key)

fmt.Println("decodeAES: ",string(decodeAES))

fmt.Println("path", r.URL.Path)

fmt.Fprintf(w, "success Hello my host!")

}

var (

    key\_t = flag.String("key\_t", "1234567890123456", "appkey for aesdecrypt")

)

func main() {

flag.Parse()

http.HandleFunc("/", sayhelloName)

err := http.ListenAndServe(":8888", nil)

if err != nil {

log.Fatal("ListenAndServe: ", err)

}

}

nodejs方法获取明文模式消息，获取数据成功回复请求服务器2XX状态码，获取数据失败回复服务器错误码可自定义，示例代码如下：

var express = require('express');

var bodyParser = require('body-parser');

var app = express();

var server = require('http').createServer(app);

app.use(bodyParser.text({

extended: false,

verify: function (req, res, buf, encoding) {

req.rawBody = buf;

}

}));

server.listen(8888);

app.post('/',function(req,res){

var body = req.body;

if(body != null){

res.send(res.statusCode + ' OK ');

var decodedstr = Buffer.from(body, 'base64');

console.log("get the decodestr : "+'\n'+ decodedstr);

}else

{

res.send(res.statusCode + " Invalid Params ");

}

res.end();

});

nodejs方法获取密文模式消息，获取数据成功回复请求服务器2XX状态码，获取数据失败回复服务器错误码可自定义，示例代码如下：

var express = require('express');

var bodyParser = require('body-parser');

var crypto = require('crypto');

var app = express();

var server = require('http').createServer(app);

app.use(bodyParser.text({

verify: function (req, res, buf, encoding) {

req.body = buf;

}

}));

var decrypt = function (key, iv, crypted) {

crypted = Buffer.from(crypted, 'base64').toString('base64');

var decipher = crypto.createDecipheriv('aes-128-cbc', key, iv);

var decoded = decipher.update(crypted, 'base64', 'utf8');

decoded += decipher.final('utf8');

return decoded;

};

server.listen(8888);

app.post('/',function(req,res){

var body = req.body;

if(body != null){

res.send(res.statusCode + ' OK ')

var crypted = body;

var key = 'F0C7E54290319978'; //添加应用产生的AppKey

var iv = 'F0C7E54290319978';

var dec = decrypt(key, iv, crypted);

console.log("数据解密后:", dec);

}else

{

res.send(res.statusCode + " Invalid Params ");

}

res.end();

});